LEARNING TO MATCH IMAGES WITH KEYPOINTS AND DESCRIPTORS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Hani Abdulaziz S Altwaijry

May 2017

LEARNING TO MATCH IMAGES WITH KEYPOINTS AND DESCRIPTORS

Hani Abdulaziz S Altwaijry, Ph.D.

Cornell University 2017

Image matching is a fundamental problem in computer vision. In the context of feature-based correspondence matching, SIFT and its variants have long excelled in a wide array of applications. Under narrow baseline viewing conditions, this problem has been successfully addressed. However, for ultra-wide baselines, as in the case of aerial images captured under large camera rotations, the appearance variation goes beyond the reach of SIFT and RANSAC. In this thesis, the problem of wide baseline matching is studied from various angles. Initially, the use of synthetic view generation and self-similarity to guide a matching procedure is leveraged to address challenges in matching aerial imagery. This is then followed by a data-driven deep-learning-based approach that sidesteps local correspondence by framing the problem as a classification task in a weak-supervision framework. However, local correspondences' usefulness is demonstrated by the incorporation of an attention mechanism to produce a set of probable matches, which allows a further increase in performance. The models outperform the state-of-the-art on ultra-wide baseline matching and approach human accuracy as per a study conducted on Amazon Mechanical Turk. Further, the learning of keypoint detection and description in a fully-supervised setting is then studied, where a large-scale dataset of patches with matching multiscale keypoints is collected. That dataset was used to learn a model capable of identifying and describing meaningful keypoints. Finally, the need for data collection is examined for the case of learning feature descriptors, where the feasibility of learning patch descriptors from synthesized viewpoint changes of random patches is investigated. The research demonstrates the effectiveness

of synthetic data in achieving comparable state-of-the-art performance on real-world non-synthetic images.

BIOGRAPHICAL SKETCH

Hani Altwaijry was born and raised in Riyadh, Saudi Arabia. He studied Computer Science and received his B.Sc. from King Saud University in 2009. He then joined King Abdulaziz City for Science and Technology as a researcher in the National Satellite Technology Program, when he was awarded a scholarship to pursue his graduate studies in the United States of America. In 2010, he attended the University of California at San Diego and completed his M.Sc. in Computer Science and Engineering in 2012. Afterward, he continued by joining the Ph.D. program there under the supervision of Prof. Serge Belongie. In 2014 he transferred to Cornell University to continue his Ph.D. research with Prof. Belongie.

To my beloved parents, Aljawharah Alkhayyal and Abdulaziz Altwaijry.

ACKNOWLEDGMENTS

I would like to begin by expressing my thanks and gratitude to my advisor Prof. Serge Belongie for his guidance and support throughout my graduate studies at both Cornell University and the University of California, San Diego (UCSD). I learned a great deal from him as he once told me how research can be a very rewarding experience. His confidence in me and encouragement, allowed me to identify and approach interesting problems and draw value from them. I would also like to express my gratitude to Prof. Mor Namaan and Prof. Noah Snavely for their valuable input and for finding the time to join my dissertation committee.

To all my wonderful friends and colleagues at the SE(3) vision group at Cornell University, words of thanks do not suffice. I would like to thank Andreas Veit, Michael Wilber, Mohammad Moghimi, Tsung-Yi Lin, and Yin Cui for all the good times. I would also like to thank my previous colleagues of the SO(3) group at the University of California at San Diego, Iljung Sam Kwak, Oscar Beijbom, Kai Wang, Steve Branson, Catherine Wah, and Eric Christensen.

I extend my thankfulness to King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia, for providing me with generous Master and Ph.D. Fellowships that allowed me to pursue my graduate studies.

My years in New York and San Diego will be forever with me, and for them I would like to thank, my brother Hotham, his wife Hana, and friends Tarfah Alrashed, Sattam Alsubaiee, Ali Alrajhi, Abdulaziz Alhussien and Hashem Alayed for all the good times while I was in San Diego, and friends at Cornell, Dipendra Misra, Akshay Bhat, Kevin Matzen and Sarah Alabdullatif.

A very special thanks to my dear friends Abdulaziz Bayounis, Jihad Alhammad, and Hossam Alfalih for their moral support and all the laughs we have

shared during those years.

I would especially like to thank my brothers and sisters, Hesham, Haithem, Najla, Najwa, Nojood, Hotham (again) and (my only little sister) Najd for being the best brothers and sisters anyone could hope for or imagine.

Throughout my Ph.D. years, my wife Nouf has been a tremendous source of strength with all her love, support, and sacrifice. Words cannot contain how full of gratitude I am to her, and I am truly lucky to have her in my life.

Finally, and above all else, my parents deserve my deepest gratitude as I am forever indebted to them. My father Abdulaziz has always been the beacon of light throughout my life. His guidance and support have been immeasurable and to him, I attribute my passion for science and technology. My mother Al-jawharah has always been my guardian angel watching over me at each and every step. Her love, unconditional; her care, limitless, have had an everlasting effect on my life. I would not be the man I am today without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Humans are predominantly visual beings. We are able to understand and relate images and their contents easily and intuitively. Image matching is a computer vision task where finding relationships between images is the primary goal and is considered a fundamental problem in the field. These relationships can be examined at different granularities. At a coarse level, we can ask whether two images show the same scene. At the other extreme, we would like to know the dense pixel-to-pixel relationships, between the two images. These granularities are directly related to broader topics in computer vision; in particular, one can look at the coarse-grained problem as an object recognition/classification task, whereas the pixel-wise problem can be viewed as one of segmentation.

Relating parts of images, known as patches, lives at the heart of many image matching approaches. Patches are defined by groupings of pixels that typically conform to certain geometrical shapes, such as squares or circles. Numerous papers have been written to describe approaches for finding suitable patches or arguing for certain patch shapes. Others are concerned with methods of encoding patch contents. Further, other works use available methods of finding and describing patches and are concerned with modeling structures in images based on aggregations of patches to accomplish higher-level tasks, such as object recognition.

This dissertation is concerned with studying problems within the domain of finding and describing patches for purposes of matching correspondences across image pairs. We explore approaches for relating images taken with a large baseline separation. Furthermore, we study approaches for feature learning with goals in identifying salient image regions and invariant encodings of

patch contents.

Chapter 2 defines the problem of ultra-wide baseline image matching in aerial images. Primarily motivated by the plethora of online mapping services available, such as Google Maps [28] and Bing Maps [22], oblique aerial imagery is now accessible to everyone. As humans, we do not have any difficulty in relating images taken with a 90° azimuth change, whereas computer vision techniques typically fail due to perspective distortions, occlusions, and lighting variability. We propose an approach that directly addresses the distortion effects by simulating viewpoint changes and resolves issues in matching repeating patterns by capitalizing on their existence to drive a matching procedure that relates the simulated viewpoint changes and selects the best geometrically meaningful match.

Chapter 3 continues to explore the problem of ultra-wide baseline image matching under a different lens. It expands upon Chapter 2 by introducing a large dataset of 50,000 matching aerial image pairs. Those 50,000 pairs are not annotated with detailed coordinate matches, however. We address the problem by introducing a model that learns to match images solely based on their contents, and further proposes regions where the image contents are likely to be in correspondence. The approach leverages convolutional neural networks equipped with attention modules to learn how to relate images globally and locally with weak supervision.

Chapter 4 shares the spirit of both chapters 2 and 3 by studying the problem of finding and encoding patches in aerial images in a learning framework based on neural networks and full supervision. To enable the fully supervised framework, we collect more oblique aerial imagery, with the goal of using existing structure-from-motion methods to construct a supervised dataset of patch-level

coordinate matches. We use the newly constructed dataset to learn a multi-scale keypoint detector and a robust patch descriptor.

Chapter 5 examines the need for data collection as we previously did in Chapter 4. A synthetic data generation approach is proposed for simulating viewpoint changes, as previously done in Chapter 2. The synthetic data is then demonstrated to allow a neural network to achieve state-of-the-art performance on patch-matching datasets.

Chapter 6 concludes this dissertation by discussing future directions for this research. Neural networks are expanding the horizon on patch detectors and descriptors performance. We believe that by studying proper data acquisition and generation techniques, and state-of-the-art neural architectures the performance of image matching pipelines can be boosted further enabling better performance for higher-level vision tasks.

CHAPTER 2

MATCHING ULTRA-WIDE BASELINE AERIAL IMAGES

Today, a large amount of aerial imagery is available online via mapping services such as Google Maps [28] or Bing [22]. These images are typically tied to location and orientation metadata. If we were to pick any pair of images from two different aerial views (see Figure 2.1 for an example), and perform SIFT-based [45] correspondence matching, we would find ourselves with a large number of mismatches due to the large distortions between the images. Even when augmenting these methods with robust approaches such as RANSAC [26] and its variants, we would still fail at finding correct correspondences since RANSAC has difficulty calculating the correct model without a large ratio of correct matches to outliers. These difficulties – large distortions, and a low ratio of correct matches to outliers – together render traditional methods ineffective. This problem has been called "Ultra-wide" baseline correspondence matching because the distance and angle from which these two images were taken is extremely large and cannot be explained by small translations or rotations [66].

In this chapter, we consider the problem of correspondence matching for aerial imagery in urban environments. Our approach builds on multiple ideas

(a) (b)

Figure 2.1: Two pairs of aerial images with correspondences are shown. Notice the large affine transforms and repeated structure exhibited in the two, as well as the varying lighting conditions.

in the literature. Namely, A-SIFT [80], patch-based methods [63], Generalized RANSAC framework [83], self-similarity [60, 32], graph-based image matching [37], and geometric-invariance [50]. The main idea behind this work is to combine view-synthesis with multiple point correspondences under an RANSAC-based scheme. Robust model estimation is supported by self-similarity principles and graph-based modeling that drives the sampling process in a restricted manner that allows the correct model to be extracted. Each of these ideas was chosen to deal with specific problems that cause failures in the earlier approaches as we will now briefly describe.

Synthesis vs. Normalization: As described in [80], features usually employ two techniques to achieve certain invariance properties. Those two techniques are synthesis and normalization. In the first case, different possibilities are synthesized to make up for certain changes. For example, in A-SIFT [80] and in [38], different affine transformations were synthesized to capture appearance changes. However, when normalization is used, the calculated feature is projected to some nominal standard, which can be difficult to produce, such that different instances could be projected to that same standard. We believe the ultra-wide baseline nature in aerial images calls for view synthesis, and hence, we follow in the footsteps of A-SIFT and adopt affine synthesis.

Patches: In feature-based approaches, a detector is implemented to find points or regions that are salient. A descriptor is then built by using a support region around given keypoints. In the case of aerial imagery, the images exhibit similar scale that allows us to disregard scale changes to a certain degree. Therefore, a fixed-size patch is likely to yield good results under this assumption, especially when augmented with affine transforms that include small-scale changes.

Multiple-Correspondence RANSAC: In the Generalized RANSAC framework of [83], multiple point correspondences are allowed by having points that satisfy a distance threshold as viable candidate matches, as opposed to a match uniqueness criteria as with the SIFT [45]. By allowing multiple correspondences we overcome the case of repeated structure, however, it gives rise to ambiguities that need to be resolved. When we incorporate view-synthesis to the system, a combinatorial explosion of possibilities arise. This requires more guided sampling for RANSAC.

Self-similarity and graph-based representation: In [32], textures comprising repeated elements were detected by correlating regions around different keypoints with each other. In a similar sense, repeated structure also arises in buildings' facades. This signals the need for a method to disambiguate our possible matches. We see a number of graph-based approaches [37, 10] used in image correspondence matching. We connect these two ideas by creating a graph of self-similar patches in our images which we use to drive the Multiple-Correspondence RANSAC sampling process.

2.0.1 Dataset

We collected 30 aerial image pairs showing buildings from different aerial vantage points from Google Maps [28]. As far as we know, there are no previous datasets dedicated for ultra-wide baseline aerial imagery. The examples were hand picked to be representative for most aerial scenes of urban environments, and such that buildings exhibit a dominant plane.

2.1 Related Work

2.1.1 Correspondence Matching

Scale-invariant Feature Transform (SIFT) [45] presented a large step in feature-based matching. A large body of work has appeared since then, including many other feature descriptors such as SURF [9], BRISK [39], and FREAK [52]. These feature descriptors usually perform badly under extreme viewpoint changes, leading to failure even when applied in an RANSAC framework.

A-SIFT [80] integrates affine-invariance to SIFT by synthesizing affine views of the two scenes under consideration. The different synthesized images are then passed through the standard SIFT keypoint detection and description process. While this approach sounds applicable to our problem, the huge number of matches and the ambiguous repeating structures defeat the approach. In A-SIFT, the affine transformations applied to the images are discarded after extracting descriptors. This leads to a heavy dependence on the matching and robust estimation approach because random sampling cannot be prevented from mixing different affine transformations in a local region.

D-Nets [74] take a different approach in finding correspondences. Their method generates lines between keypoints or grid points and calculates descriptors for each line. The line segments from the two images are matched through a hashing and voting scheme. Their method delivers both good performance and accuracy. Their departure from conventional patch-based approaches offers good insight into correspondence matching and therefore we compare our approach to D-Nets.

2.1.2 Ultra-wide Baseline Matching

There have been several works in wide baseline stereo matching [54, 70]. However, in those cases, the distortions exhibited in the pair of images are not very large. For the case of "Ultra-wide" baseline matching, several works have been presented.

The "scale-selective self-similarity" S^4 descriptor was presented by Bansal *et al*. [8] which they used in performing geolocalization of street view images through facade matching against labeled bird's eye view aerial images. In their case, the aerial images were labeled by marking the existing building facades for rectification purposes.

Chung *et al*. [20] present a method for building recognition by employing semantically rich sketch representations that are matched with a spectral graph matching procedure. Their method uses MSERs [46] to detect affine-regions that are then used to find repeating structure, which in turn are used to create a sketch representation. One interesting aspect of their work is their use of geometrical invariants based on node relationships. Our method shares the spirit of this approach, as we will describe shortly.

In [84], Zhang *et al*. present a visual-phrases approach to image retrieval. It is supported by imposing geometric constraints over the different visual words in a given scene. The geometry preserving notions they present highlights the importance of respecting geometry between keypoints occurring together spatially.

2.1.3 Robust Estimation

Correspondence matching is often supported by robust estimation approaches such as RANSAC [26] to extract the correct model representing the underlying

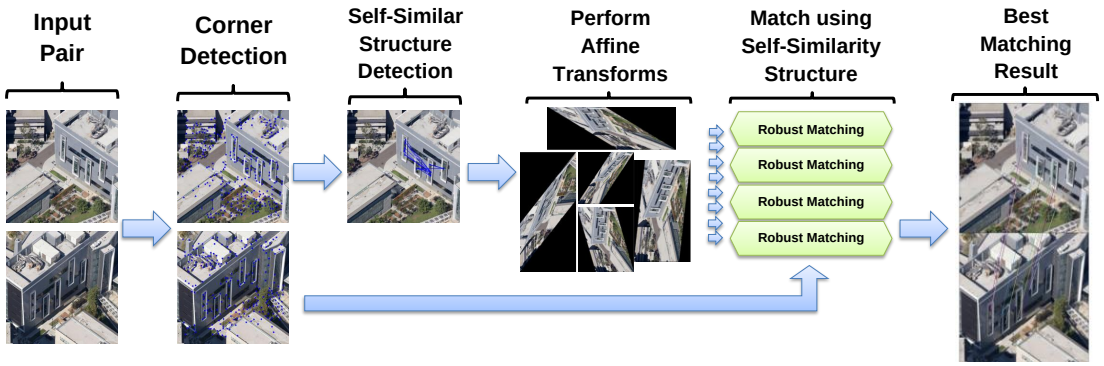


Figure 2.2: An overview of the matching pipeline is shown here. The details of the approach are discussed in Section 2.2.

geometry. Many variants of RANSAC exist to solve different problems, such as the existence of multiple models (Multi-RANSAC [86], Sequential-RANSAC [73]) as in multiple facets of a building. In these variants, the notions of multiple point correspondences are not considered. Other variants such as PROSAC [19] perform guided sampling to increase robustness to outliers.

The Generalized RANSAC framework [83], incorporates the notion of many-to-many matching in RANSAC as an effort to overcome repeated structure or self-similarity problems. However, it still based on random sampling which does not respect spatial structure. This leads to many draws that give rise to incorrect models.

In the literature, there are other approaches to performing a matching under the many-to-many paradigm. Namely, spectral methods such as [15] and optimization based methods such as [17], however, space limitations do not permit their discussion.

(a) (b)

Figure 2.3: In (a), detected keypoints are shown along a subset of the patches representing them. We can see that the extracted keypoints represent good corners that are likely to be encountered in another view of the building. In (b), a sample of affine transformed patches corresponding to the keypoint shown on the right. A correctly matching patch is highlighted with a green borderline.

2.2 Approach

2.2.1 Feature Extraction and Description

For keypoint detection, we employed the standard Harris corner detection procedure [30] after smoothing the image with a Gaussian kernel. Our goal was to obtain the corner points covering most of the features on building facets.

We describe our keypoints by placing a window of size $p \times p$ around each keypoint making a square patch, and then we compute the Histogram of Oriented Gradients (HOG) [23]. Our use of HOG was due to its power of capturing the gradient structure and its wide success in the object recognition literature. Figure 2.3(a) shows an example of detected keypoints and sample patches.

2.2.2 Affine Synthesis

The aerial imagery under consideration seems to obey the affine camera model to some extent, as the camera is very distant from the imaged objects, and the field-of-view is small. This leads us assume affine local regions, and there-

fore following in the spirit of A-SIFT [80], we synthesize affine transformations. However in A-SIFT, the transformations are applied to both input pairs, and follows a different sampling procedure. We apply our transformations to one of the input pairs only, and as follows:

$$Scale = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} Shear = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} Rotation = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\forall S_{x,y} \in [S_{begin} : S_{step} : S_{end}], \forall Sh_{x,y} \in [Sh_{begin} : Sh_{step} : Sh_{end}], \forall \theta \in [\theta_{start} : \theta_{step} : \theta_{end}] \tag{2.2}$$

$$A = Scale \times Shear \times Rotation, \qquad I_{S_x,S_y,Sh_x,Sh_y,\theta} = A \times I \tag{2.3}$$

where I is an image.

The transformations applied belong to a subset of the affine transformations group. The different variable ranges for, *e.g.* $S_{x,y}$, are chosen to cover a wide variety of affine transformations that should capture the expected distortions in the aerial imagery. Figure 2.3(b) shows instances of affine transformed patches, and a corresponding patch from the target image.

2.2.3 Self-Similarity Graph

Buildings, in general, exhibit features that are similar to one another which is due to architectural designs with repeating patterns of windows, balconies, railings, etc. This is leveraged by forming a graph over similar patches in one of the input images. Note that we only consider one image from the input pair for the

self-similarity information and not both. The reason will become clear during the matching stage.

We begin our self-similarity computation by calculating the distance matrix D for all pairs of patches by comparing their HOG descriptors using the l_2 norm, i.e.:

$$D_{ij} = ||h_i - h_j||_2 \tag{2.4}$$

where h_i is the HOG descriptor of the patch i. Afterwards, we proceed by constructing a graph $G(V, E)$ with the adjacency matrix M, such that:

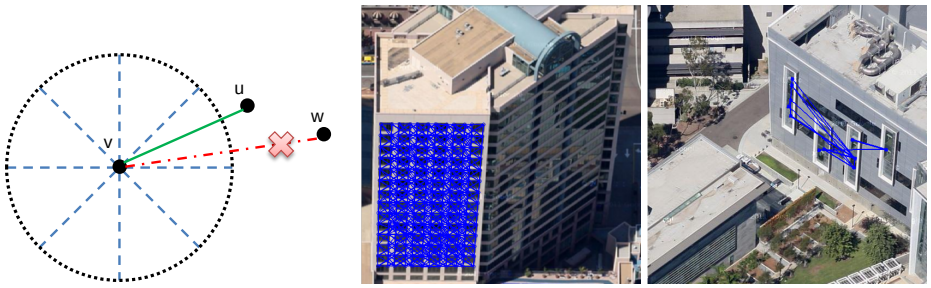
$$M_{ij} = \begin{cases} 1 & \text{if } D_{ij} \leq \tau_1, i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where τ_1 is a distance threshold. Using the adjacency matrix M, we find all connected components $C_i(V', E')$ such that:

$$\forall v, u \in V' \iff \text{a path exists between } v \text{ and } u \tag{2.6}$$

After finding all connected components within G, we select the connected component with the largest cardinality of vertices after passing a non-collinearity test. Then, we simplify it by introducing geometric relations. First, around each vertex, we divide the space into k angular bins. A vertex $v \in V'$ is allowed to have up to k neighbors, such that an angular bin can only have a single neighbor u. We select u as the geometrically closest neighbor to v falling into that angular bin. An example of this step is illustrated in Figure 2.4(a).

The result of this step is a connected component that describes the structure of self-similar patches. A sample of a simplified self-similar structure is shown in Figure 2.4(b).



(a) (b)

Figure 2.4: In (a), we illustrate the angular binning around a given vertex v, and show how we assign the vertex u as the appropriate neighbor, as opposed to choosing the vertex w. The choice is made based on geometrical distance. In (b), two examples of the largest connected component in its simplified form using 9 angular bins.

Figure 2.5: An instance of a transformed input image, and the sampled minimum set. Although the correspondences shown are incorrect in this instance, the spatial configuration is respected, which is the goal behind using the self-similar graph sampling strategy.

2.2.4 Matching and Robust Model Estimation

When proceeding to the matching stage, approaches like A-SIFT [80] discard the affine transformations they used. We believe that if two patches match as caused by affine transforming one of them, then this affine transform gives us hints about the underlying local geometry that could lead to such matching. Therefore, we explicitly incorporate our affine transformations as part of our RANSAC-based robust estimation method.

Let us call our input images I_1 and I_2. We begin by calculating keypoints on both of I_1 and I_2, and the self-similarity graph obtaining the connected compo-

Algorithm 1 Affine Synthesis

Require: Affine Transforms A^*, Connected Component C^*, Patch Set P_1, Patch Set P_2

$AllModels \leftarrow \phi$

for $A \in A^*$ **do**

$P_1' \leftarrow ApplyAffine(A, P_1)$

$C \leftarrow ApplyAffine(A, C^*)$

$h_i \leftarrow HOG(p_i)\ \forall p_i \in P_1'$

$D_{ij} \leftarrow ||h_i - h_j||_2, \forall p_i \in P_1', p_j \in P_2$

$S \leftarrow \{(i,j) : D_{ij} \leq \tau_2\}$

$NewModel \leftarrow RANSAC(S, C, P_1', P_2, nIter)$ // Execute RANSAC Algorithm 2

$AllModels \leftarrow AllModels \cup NewModel$

end for

return $BestSet$:= $max_{AllModels}\ |BestSet|$

nent C^* from I_1. Let P_1 be the set of patches defined by the vertices of C^*, and let P_2 be all patches from I_2. We proceed with Algorithm 1, which applies all affine transformations under consideration to the input data. When an affine transform A is applied, we calculate our matches and transfer control to Algorithm 2, which is a Multiple-Correspondence RANSAC that samples the data according to the transformed connected component C.

In its essence, the algorithm samples points in the input pair that respect a certain spatial configuration. That configuration ensures that points sampled in the transformed I_1, and in the target I_2 will have *the same geometric relationship*. This enforcement is achieved by maintaining the angular binning relationships of the pairs of points in the current sample. As a result, this decreases the number of random samples to be taken as opposed to randomly picking correspondences. An example of a sample following geometric constraints is shown in Figure 2.5.

Currently, a single homography is estimated, which is clearly a limitation. However, for an initial test of our approach, we believe this is sufficient as we

aim to capture the dominant plane in the scene. A final note on our implementation, the best homography guess is passed through a final RANSAC round seeded with the best homography. If RANSAC produced a larger consensus set, we choose the new model, otherwise, we keep the older one. This seemed to increase the robustness of the estimation.

The algorithm performs $O(nIter \cdot |A^*|)$ RANSAC runs, and in each run, it performs $O(nm + |C^*||S| + n)$ operations where nm account for matching n points from I_1 with m points in I_2, and $|C^*||S|$ account for worst case neighbor matching, and finally n for model evaluation.

Algorithm 2 RANSAC with Graph Sampling

Require: Match Set S, Connected Component C^*,
Point Set P_1, Point Set P_2,
RANSAC Iteration Count $nIter$

$BestModel \leftarrow \phi$, $BestSet \leftarrow \phi$
for $iter \leq nIter$ **do**
  Pick $v \in C^*$ randomly
  Pick $v' \in P_2$, such that $(v, v') \in S$
  $MinSample \leftarrow$ NeighborMatch(v, v', $[v\ v']$)
  $Homography \leftarrow HomographyDLT(MinSample)$
  $ConSet \leftarrow$ EvaluateModel(Homography, P_1, P_2)
  if $|ConSet| > |BestSet|$ **then**
    $BestModel \leftarrow Homography$
    $BestSet \leftarrow ConSet$
  end if
  return $(BestModel, BestSet)$
end for

```
function NEIGHBORMATCH(v, v′, Q)
    /* The goal of neighbor match is to find
    correspondences that exhibit the same spatial
    layout by looking at matching angle bins*/
    loop    // over neighbors of v
        Pick u ∈ C*, such that (v, u) ∈ C*
        Pick u′ ∈ P2, such that (u, u′) ∈ S
        if angleBin(v,u) = angleBin(v′,u′) then
            if |Q|≤ minCount then
                return NeighborMatch(u,u′, Q ∪ [u u′])
            else
                return Q ∪ [u u′]
            end if
        end if
    end loop
    return FAILURE
end function
```

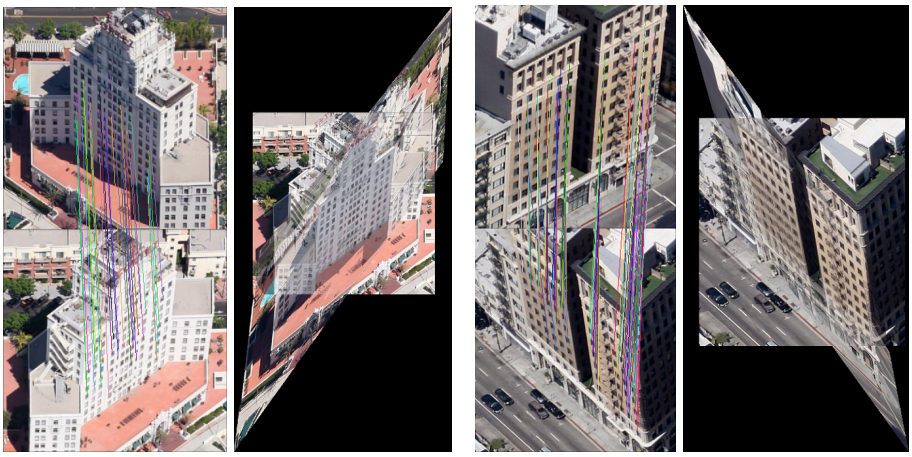
2.3 Experiments

2.3.1 Implementation Details

In our implementation, we used a patch size of 50×50. Each cell in the HOG descriptor covered 5×5 pixels. The number of iterations $nIter$ is set to 5000. The affine transforms ranges were chosen reasonably to cover possible transformations occurring in the aerial imagery. Certain assumptions were made when choosing these values, *e.g.* we do not have 90-degree 2D-rotations present in the aerial imagery. We ran our implementation in two configurations, to measure its sensitivity to parameter changes. Their details are as follows:

In the first configuration we set $\tau_1 = 6$, $\tau_2 = 7$. The scale factors were chosen as $S_{x,y} \in [0.5, 2]$. The shear factors were chosen as $Sh_{x,y} \in [-1.5, 1.5]$. The rotation angles were between $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$. The blur kernel was 3×3, with $\sigma = 0.4$. The Harris detector had a window size of 7×7, and a threshold of 0.001.

In the second configuration we set $\tau_1 = 6.5$, $\tau_2 = 7$. The scale factors were



(a) (b)

Figure 2.6: An example pair is matched using our method, and the recovered homography is used to stitch the two images together.

chosen as $S_{x,y} = 1$. The shear factors were chosen as $Sh_{x,y} \in [-1.75, 1.75]$. The rotation angles were between $\theta \in [-\frac{\pi}{12}, \frac{\pi}{12}]$. The blur kernel was 3×3, with $\sigma = 1.5$. The Harris detector, had a window size of 7×7, and a threshold of 0.01.

2.3.2 Experimental Setup and Results

To evaluate our approach, we input each pair of the aerial images to: (1) Our approach, (2) A-SIFT, and (3) D-Nets. For A-SIFT [80], we used the implementation provided on their website with a slight modification to estimate a homography using OpenCV [11] instead of a fundamental matrix. We believe that A-SIFT encapsulates SIFT by definition, and therefore we do not compare with the standard SIFT. For D-Nets [74], we use the implementation provided on their website in a straight forward manner employing the FAST keypoint detector.

We measure whether each of these methods finds the correct homography, or finds a shifted version of the correct one, or finds a correct but different plane, or completely fails. The results are shown in Table 2.1. A correct homography is tested against a human labeled homography and is considered correct if the

Method	Correct Homography	Shifted Homography	Different Plane	Failure	Success Rate
Ours-1	3	5	1	21	30%
Ours-2	5	4	1	20	33%
A-SIFT	1	0	5	24	20%
D-Nets	4	3	2	21	30%

Table 2.1: Results of finding homographies using two configurations of Our Approach, A-SIFT, and D-Nets.

number of correct matches exceeds 75%. Shifted versions and other planes are judged empirically using visualizations. Figure 2.6(a) shows the result of our matching algorithm on two pairs, and a visualization of the recovered homography by stitching the two images. [1]

Between the two runs, there were 7 unique correct homographies. We see that our method finds a lot of shifted homographies, especially in the cases with numerous repeated structures. In these cases, the typical cause is not finding corresponding keypoints due to the Harris threshold, or too few iterations.

Relative to D-Nets, the results are highly comparable. The issue becomes computational cost vs. memory cost. Our method is computationally intensive. On the other hand, D-Nets requires a lot of memory; they recommend about 32GB of RAM. Our Matlab implementation occupies about 0.8 GB of RAM when running, which can be greatly reduced under a different language implementation. The machine we used had a 3.4 GHz Intel Core i7 processor with 12 GB of RAM.

The failure cases we exhibit are mainly due to two main issues: (1) a self-similar connected component is not found, or poorly constructed with collinearity issues. (2) keypoints are not detected properly due to image blur. Therefore factors such as the size of the employed Gaussian blur, the Harris threshold, or

[1] Visual examples of all pairs are shown in the supplementary material.

HOG distance threshold have a great impact on the performance. We believe performance can be greatly enhanced by tweaking the connected component discovery by introducing similarity-transitivity resulting in strongly connected components that suffer fewer collinearity issues, which improves the sampling.

2.4 Conclusion

In conclusion, our proposed approach provides a step forward in the challenging real world problem of ultra-wide baseline image matching for urban environments. Through our use of affine synthesis along with the self-similarity graph, we greatly reduce the number of RANSAC iterations needed to find a solution. In our future work, we will pursue the following improvements: (1) reducing the number of affine transformations needed, (2) improving the graph operations, (3) improving the angular binning approach by including distance bins, and (4) including the support of multiple planes.

CHAPTER 3

LEARNING TO MATCH AERIAL IMAGES

Oblique aerial images acquired by distant cameras from very different angles, as shown in Fig. 3.1, are challenging for geometry-based approaches, as we have seen in Chapter 2, for a number of reasons—chief among them are dramatic appearance distortions due to viewpoint changes and ambiguities due to repetitive structures. This renders methods based on local correspondence insufficient for ultra-wide baseline matching.

In this chapter, we follow a data-driven approach. Specifically, we treat the problem from a recognition standpoint, without appealing specifically to handcrafted, feature-based approaches or their underlying geometry. Our aim is to learn a discriminative representation from numerous instances of *same* and *different* pairs, which separates the genuine matches from the impostors.

We propose two architectures based on Convolutional Neural Networks (CNNs). The first architecture is only concerned with learning to discriminate image pairs as *same* or *different*. The second one extends it by incorporating a Spatial Transformer module [33] to propose *possible* matching regions, in addition to the classification task. We learn both networks given only *same* and *different* pairs, *i.e.*, we learn the spatial transformations in a semi-supervised manner.

To train and validate our models, we use a dataset with 49k ultra-wide baseline pairs of aerial images compiled from Google Maps specifically for this problem: example pairs are shown in Fig. 3.2. We benchmark our models against multiple baselines, including human annotations, and demonstrate state-of-the-art performance, close to that of the human annotations.

Our main contributions are as follows. **First,** we demonstrate that deep CNNs offer a solution for ultra-wide baseline matching. Inspired by recent ef-

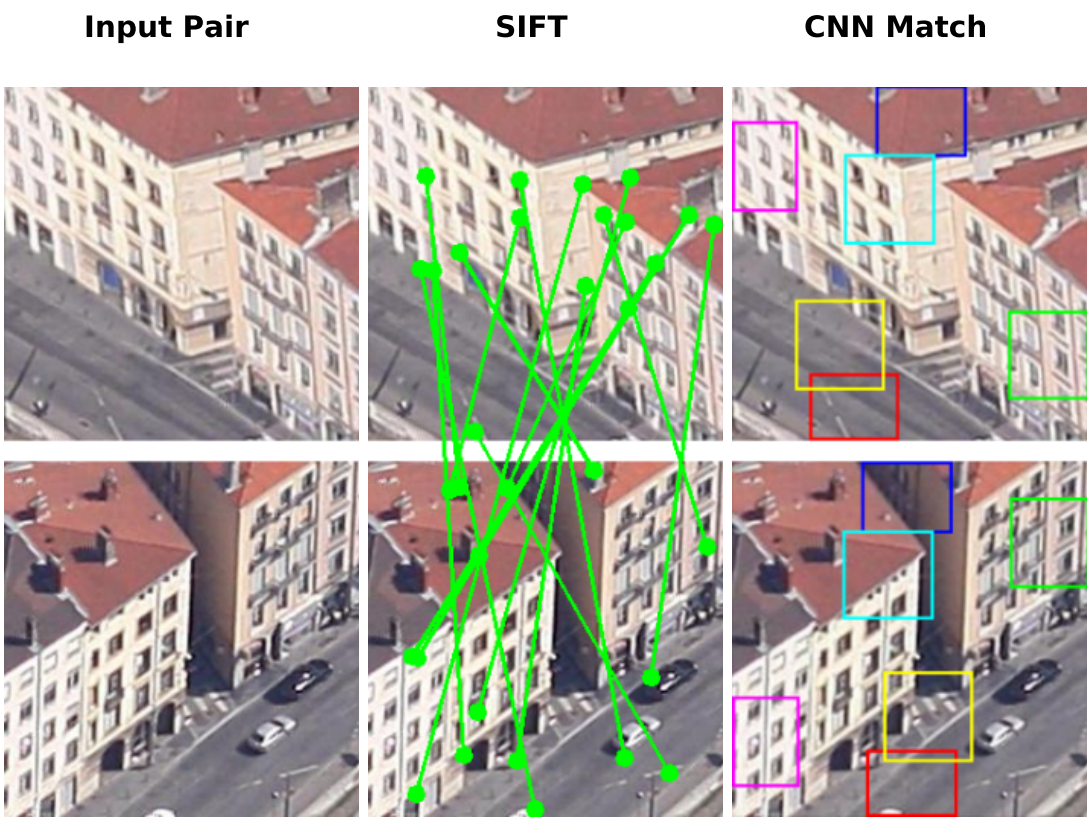


Figure 3.1: Matching ultra-wide baseline aerial images. Left: The pair of images in question. Middle: Local correspondence matching approaches fail to handle this baseline and rotation. Right: The CNN matches the pair and proposes possible region matches.

forts in patch matching [29, 81, 61] we build a siamese/classification hybrid model using two AlexNet networks [36], cut off at the last pooling layer. The networks share weights and are followed by a number of fully-connected layers embodying a binary classifier. **Second,** we show how to extend the previous model with a Spatial Transformer (ST) module, which embodies an attention mechanism that allows our model to propose *possible* patch matches (see Fig. 3.1), which in turn increases performance. These patches are described and compared with MatchNet [29]. As with the first model, we train this network end-to-end, and only with *same* and *different* training signal, *i.e.*, the ST module is trained in a semi-supervised manner. In sections 3.2.2 and 3.3.6 we discuss the

Figure 3.2: Sample pairs from one of our datasets, collected from Google Maps [28] 'Birds-Eye' view. Pairs show an area or building from two widely separated viewpoints.

difficulties in training this network, and offer insights in this direction. **Third,** we conduct a human study to help us characterize the problem, and benchmark our algorithms against human performance. This experiment was conducted on Amazon Mechanical Turk, where participants were shown pairs of images from our dataset. The results confirm that humans perform exceptionally while responding relatively quickly. Our top-performing model falls within 1% of human accuracy.

3.1 Related Work

3.1.1 Correspondence Matching

Correspondence matching has been long dominated by feature-based methods, led by SIFT [45]. Numerous descriptors have been developed within the community, such as SURF [9], BRIEF [14], and DAISY [67]. These descriptors generally provide excellent performance in narrow baselines, but are unable to handle

the large distortions present in ultra-wide baseline matching [47].

Sparse matching techniques typically begin by extracting keypoints, *e.g.*, Harris Corners [30]; followed by a description step, *e.g.*, computing SIFT descriptors; then a keypoint matching step, which gives us a pool of probable keypoint matches. These are then fed into a model estimation technique, *e.g.*, RANSAC [26] with a homography model. This pipeline assumes certain limitations and demands assumptions to be made. Relying on keypoints can be limiting—dense techniques have been successful in wide-baseline stereo with calibration data [67, 69, 75], scene alignment [42, 75] and large displacement motion [69, 75].

The descriptor embodies assumptions about the topology of the scene, *e.g.*, SIFT is not robust against affine distortions, a problem addressed by Affine-SIFT [80]. Further assumptions are made in the matching step: do we consider only unique keypoint matches? What about repetitive structures? Finally, the robust model estimation step is expected to tease out a correct geometric model. We believe that these assumptions play a major role in why feature-based approaches are currently incapable of matching images across very wide baselines.

3.1.2 Ultra-wide Baseline Feature-Based Matching

Ultra-wide baseline matching generally falls under the umbrella of correspondence matching problems. There have been several works on wide-baseline matching [66, 46]. For urban scenery, Bansal *et al*. [8] presented the Scale-Selective Self-Similarity (S^4) descriptor which they used to identify and match building facades for image geo-localization purposes. In Chapter 2, we matched urban imagery under ultra-wide baseline conditions with an approach involving affine invariance and a controlled matching step. Chung *et al*. [20] calculate

sketch-like representations of buildings used for recognition and matching. In general, these approaches suffer from poor performance due to the difficulty of the problem.

3.1.3 Convolutional Neural Networks

Neural Networks have a long history in the field of Artificial Intelligence, starting with [58]. Recently, Deep Convolutional Neural Networks have achieved state-of-the-art results and become the dominant paradigm in multiple fronts of computer vision research [36, 64, 65, 27].

Several works have investigated aspects of correspondence matching with CNNs. In [44], Long *et al*. shed some light on feature localization within a CNN, and determine that features in later stages of the CNN correspond to features finer than the receptive fields they cover. Toshev and Szegedy [68] determine the pose of human bodies using CNNs in a regression framework. In their setting, the neural network is trained to regress the locations of body joints in a multi-stage process. Lin *et al*. [41] use a siamese CNN architecture to put aerial and ground images in a common embedding for ground image geo-localization.

The literature has seen a number of approaches to learning descriptors prior to neural networks. In [13], Brown *et al*. introduce three sets of matching patches obtained from structure-from-motion reconstructions and learn descriptor representations to match them better. Simonyan *et al*. [62] learn the placement of pooling regions in image-space and dimensionality reduction for descriptors. However, with the rise of CNNs, several lines of work investigated learning descriptors with deep networks. They generally rely on a two-branch structure inspired by the siamese network of [12], where two networks are given pairs of matching and non-matching patches. This is the approach followed by Han *et al*.

with MatchNet [29], which relies on a fully connected network after the siamese structure to learn the comparison metric. DeepCompare [81] uses a similar architecture and focuses on the center of the patch to increase performance. In contrast, Simo-Serra *et al*. [61] learn descriptors that can be compared with the L_2 distance, discarding the siamese network after training. These three methods relied on data from [13] to learn their representations. They assume that salient regions are already determined, and deliver a better approach to feature description for feature-based correspondence matching techniques. The question of obtaining CNN-borne correspondences between two input pairs, however, remains unexplored.

Lastly, attention models [49, 5] have been developed to recognize objects by an attention mechanism examining sub-regions of the input image sequentially. In essence, the attention mechanism embodies a saliency detector. In [33], the Spatial Transformer (ST) network was introduced as an attention mechanism capable of warping the inputs to increase recognition accuracy. In section 3.2.2 we discuss how we employ an ST module to let the network produce guesses for probable region matches.

3.2 Deep-Learning Architectures

3.2.1 Hybrid Network

We introduce an architecture which, given a pair of images, estimates the likelihood that they belong to the same scene. Inspired by the recent success of patch-matching approaches based on CNNs [81, 29, 61], we use a hybrid siamese/classification network. The network comprises two parts: two feature extraction arms that share weights (the *siamese* component) and process each

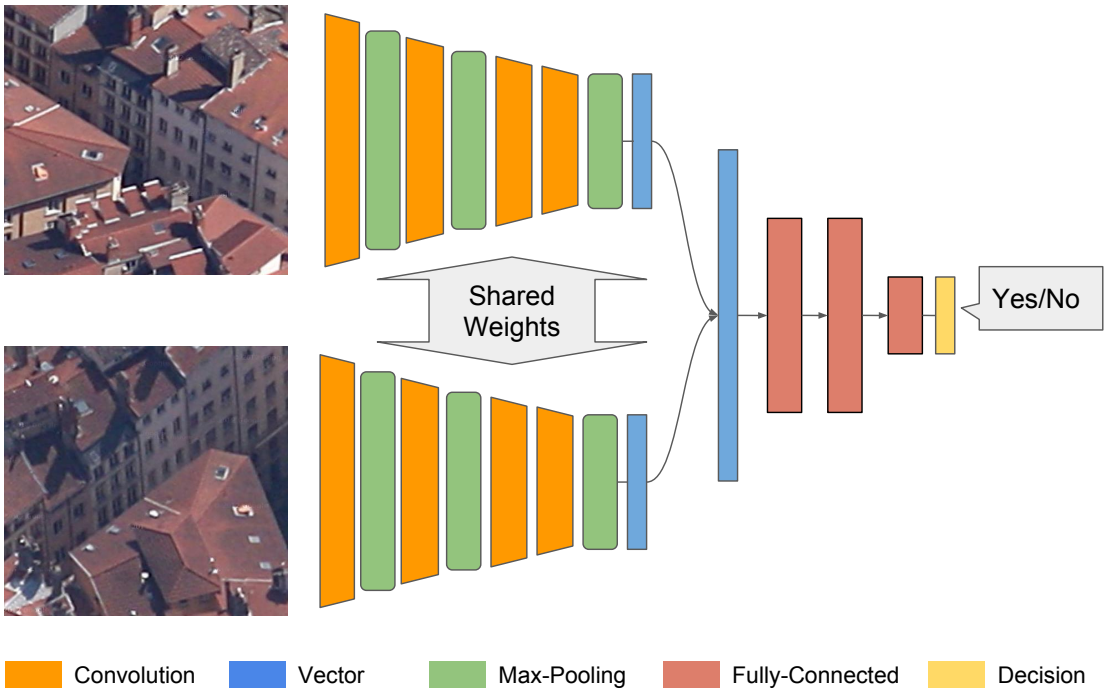


Figure 3.3: The siamese/classification Hybrid network. Weights are shared between the convolutional arms. ReLU and LRN (Local Response Normalization) layers are not shown for brevity.

input image separately, and a *classifier* component that produces the matching probability. For the siamese component, we use the convolutional part of AlexNet [36], *i.e.*, cutting off the fully connected layers. For the classifier, we use a set of fully-connected layers that takes as input the concatenation of the siamese features and ends with a binary classifier, for which we minimize the binary cross-entropy loss. Fig. 3.3 illustrates the structure of the 'Hybrid' network.

The main motivation behind this design is that it allows features with local information from both images to be considered jointly. This is achieved where the two convolutional features are concatenated. At that layer, the features from both images retain correspondence to specific regions within the input images.

3.2.2 Hybrid++

Unlike traditional geometry-based approaches, the hybrid network proposed in the previous section does not model local similarity explicitly, making it difficult to draw conclusions about corresponding image regions. We would like to determine whether modeling local similarities more explicitly can produce more discriminative models.

We, therefore, sought to expand our hybrid architecture to allow for predictions of *probable* region matches, in addition to the classification task. To accomplish this, we leverage the Spatial Transformer (ST) network described in [33]. Spatial transformers consist of a network used for localization, which takes as input the image and produces the parameters for a pre-determined transformation model (*e.g.*, translation, affine, etc.) which is used in turn to transform the image. It relies on a grid generator and a differentiable sampling kernel to keep track of the gradient propagation to the localization network. The model can be trained with standard back-propagation, unlike the attention mechanisms of [5, 49] that relied on reinforcement learning techniques. The *spatial transformer* is typically a standard CNN followed by a set of fully-connected layers with the required number of outputs, *i.e.*, the number of transformation parameters, *e.g.*, two for translation, six for affine.

The spatial transformer allows for any transformation as long as it is differentiable. However, in this work we only consider extracting patches at a fixed scale, *i.e.*, translations, which are used to generate patch proposals over both images—richer models, such as perspective transformations, can potentially be more descriptive, but are also more difficult to train.

We build the spatial transformer with the same convolutional network used for the 'arms' of the siamese component of our hybrid network, plus a set of

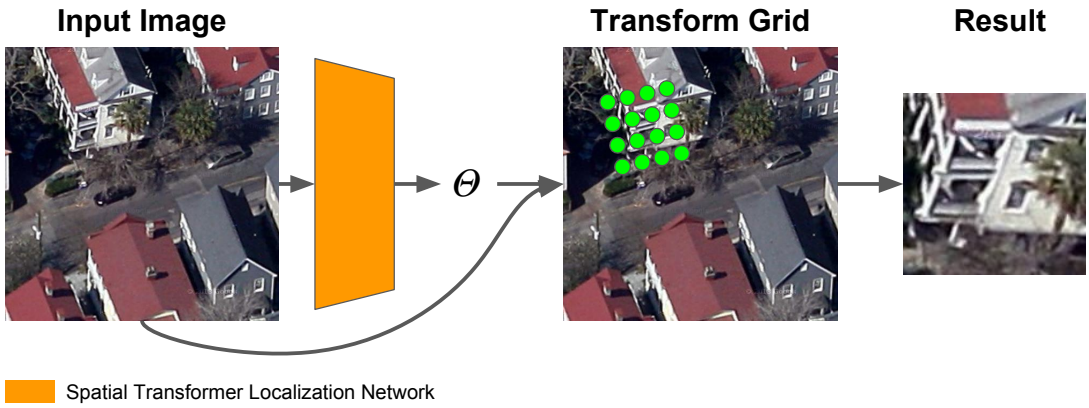


Figure 3.4: Overview of a Spatial Transformer module operating on a single image. The module uses the regressed parameters Θ to generate and sample a grid of pixels in the original image.

fully connected layers that regress the transformation parameters $\Theta = \{\Theta_1, \Theta_2\}$, which are used to transform the input images, effectively sampling patches. Note that patch locations for each individual image are a function of *both* images. The number of extracted patches is reflected in the number of regressed parameters specified. Fig. 3.4 illustrates how the spatial transformer module operates.

The spatial transformer modules allow us to explicitly model regions within each input image, permitting the network to propose similar regions given an architecture that demands such a goal. The overall structure of this model, which we call 'Hybrid++', is shown in Fig. 3.5.

Describing Patches

In our model, we pair an ST module which produces a pre-determined number of fixed-scale patch proposals with our hybrid network. The extracted patches are given to a MatchNet [29] network, which was trained with interest points from Structure-from-Motion data [13] and thus already has a measure of invariance against perspective changes built-in.

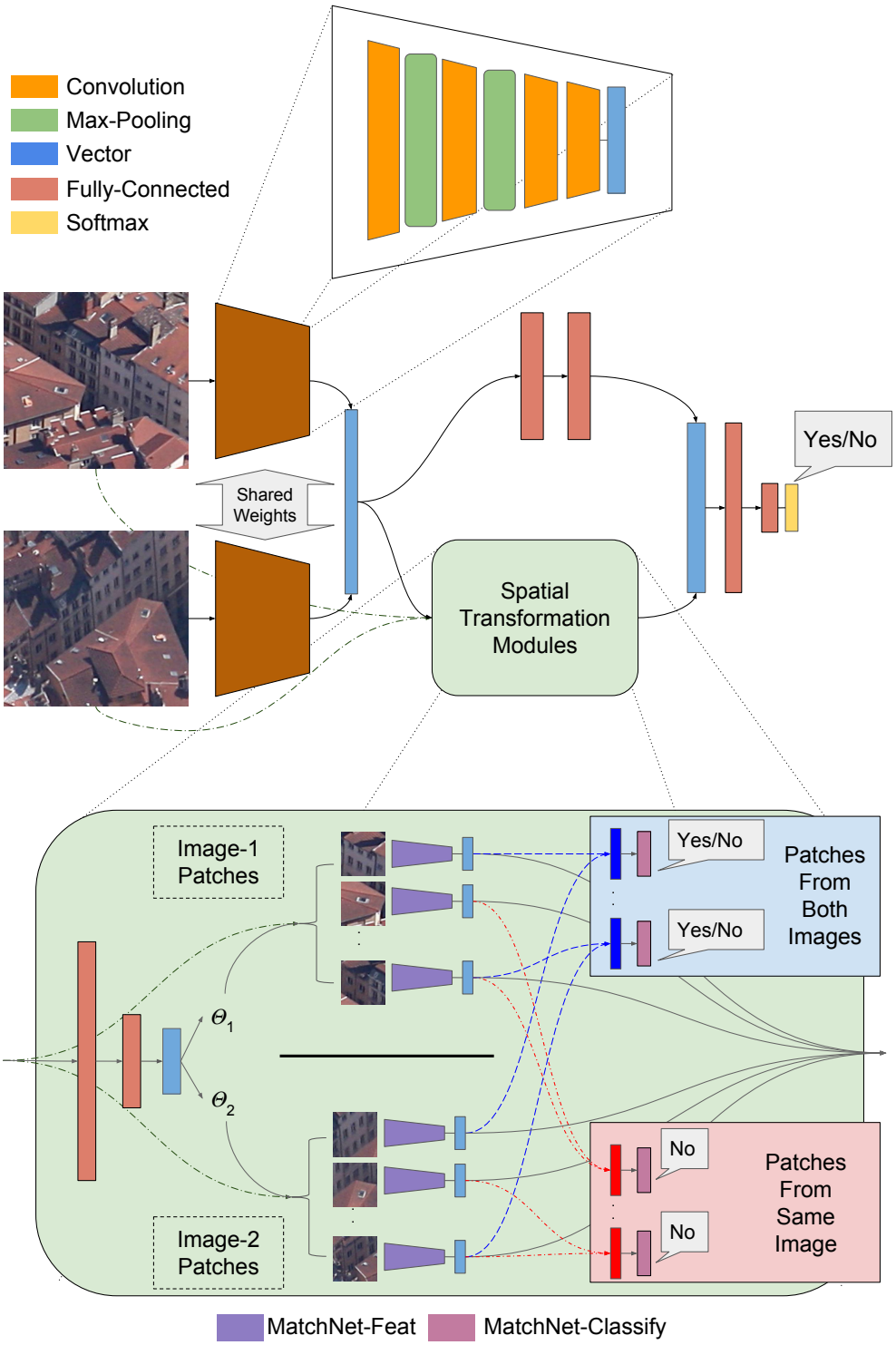


Figure 3.5: The 'Hybrid++' Network. Spatial Transformer modules are incorporated into the 'Hybrid' model to predict probable patch matches.

MatchNet has two components in its network, a feature extractor modeled as a series of convolutional layers, and a classifier network that takes the outputs of two feature extractors and produces a similarity score. We pass each extracted patch, after converting it to grayscale, through the MatchNet feature extractor network (MatchNet-Feat) and arrive at a 4096-dimensional descriptor vector.

These descriptors are then used for three different objectives. The first objective is to supplement the global feature description extracted by the original hybrid architecture. In this manner, the extracted descriptors provide the classifier with information extracted at a dedicated higher-resolution mode. The second objective is to match patches in the *other image*. This objective encourages the network to use the spatial transformer to focus on similar patches in both images simultaneously. The third objective is for the patch to *not match* other patches extracted from the *same image*, which we mainly use to discourage the network from collapsing onto a single patch. For the last two tasks, we use the MatchNet classification network (MatchNet-Classify).

Optimization

Combining the image-wise classification objective with the regional descriptor objectives yields an objective function with four components:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \Big(\mathcal{L}_{\text{class}} + \alpha \mathcal{L}_{\text{patch}} + \beta \mathcal{L}_{\text{pairwise}} + \gamma \mathcal{L}_{\text{bounds}} \Big) \tag{3.1}$$

where N is the size of the training batch and α, β, γ are used to adjust the weights. The first component of the loss function encodes the image classification objective:

$$\mathcal{L}_{\text{class}} = y_i \log p_i + (1 - y_i) \log(1 - p_i) \tag{3.2}$$

where p_i is the probability of the images matching and $y_i \in \{0, 1\}$ is the label. The second component encodes the match of each pair of patches across both

images:

$$\mathcal{L}_{\text{patch}} = \frac{1}{M} \sum_{m=1}^{M} \Big[y_i \log q_m + (1 - y_i) \log(1 - q_m) \Big] \tag{3.3}$$

where M is the number of patches, and q_m is the probability of patch $\mathbf{x}_m^1$ on image 1 matching patch $\mathbf{x}_m^2$ on image 2. The third component is a pairwise penalty function that discourages good matches among the patches within the same image, to prevent the network from collapsing the transformations on top of each other:

$$\mathcal{L}_{\text{pairwise}} = \frac{4}{M(M-1)} \sum_{t=1}^{2} \sum_{m=1}^{M} \sum_{k=m+1}^{M} \log(1 - u_{m,k}^t) \tag{3.4}$$

where $u_{m,k}^t$ is the probability of patch $\mathbf{x}_m^t$ matching patch $\mathbf{x}_k^t$ on image $t = \{1, 2\}$. The last component is a penalty function that discourages spatial transformations that fall out of bounds:

$$\mathcal{L}_{\text{bounds}} = \frac{2}{M} \sum_{t=1}^{2} \sum_{m=1}^{M} f(\mathbf{x}_m^t) \tag{3.5}$$

where $f(\mathbf{x}_m^t)$ is a function that computes the ratio of pixels sampled out of bounds for patch $\mathbf{x}_m^t$. The out-of-bounds loss term discourages the model from stepping outside the image, which may minimize the patch-matching loss, given an appropriate weight—with this penalty function we gain more control over the optimization process.

3.2.3 Training Procedure

To train the hybrid network, we follow a standard training procedure by fine-tuning the model after loading pre-trained AlexNet weights into the convolutional arms only. However, training the Hybrid++ network is more subtle, as the network needs to get started on the right foot. We initially train the non-ST and ST sides separately with the global *yes/no* matching signal. Afterward, we

train the networks jointly. We learned this is necessary to prevent the network from shutting off one side while minimizing the objective. Similar to the Hybrid case, we use pre-trained weights for the convolutional arms.

We use MatchNet as a pure feature descriptor, with frozen weights, *i.e.*, no learning. This is primarily done to prevent the network from minimizing the loss by changing the descriptors themselves without moving the attention mechanism. Our training procedure does not have pixel-to-pixel correspondence labels, and hence we do not know if the network is examining similar patches. We rely on the power provided by MatchNet to determine patch similarity. The global matching label, in turn, becomes a semi-supervised cue. Therefore, the network can only minimize the loss component for patch matching by moving the attention mechanism to examine patches that appear to be similar, as per MatchNet.

The reliance on MatchNet is a double-edged sword, as it is our only means of moving the attention mechanism without explicit knowledge of labeled patch correspondences. That means if MatchNet cannot find the correspondence for two patches that do match, then the attention mechanism cannot learn to look for these two patches.

3.3 Experiments

3.3.1 Dataset

We compiled 49,271 matching pairs (98,542 images) of oblique aerial imagery through Google Maps [28]. The images were collected using an *automated process* that looks for planar surfaces such that the normal vector of the surface is within 40° to 75° of one cardinal direction. This guarantees the visibility of the surface

from two different viewpoints. The pairs were collected non-uniformly from San Francisco, Boston, and Milan. Those locations were chosen with a goal of diversifying the scenery.

We split the dataset into roughly $\sim$39K/$\sim$10K training/testing positive pairs. For training, we generate samples in an online manner by sampling from the reservoir of positive matching pairs. The sampling procedure is set to produce samples with a 1:1 positive:negative ratio. Therefore, a random classifier would score 50% on the test set. We call this the 'aerial' dataset.

3.3.2 Human Performance

We ask ourselves: How well do humans perform when matching such images? To this end, we conducted a small experiment with human participants on Amazon Mechanical Turk [4]. We picked a subset of 1,000 pairs from our test set and presented them to the human subjects. Each participant was shown 10 pairs of different images and was asked to determine whether each pair showed the same area or building, as a binary question. We show a screenshot of the interface presented to the participants in Fig. 3.6. Each pair of images was presented at least 5 times to different participants, giving us a total of 5000 labels, 5 per pair.

Our interface was prone to adversarial participants, those answering randomly or giving a constant answer all the time. To mitigate the effect of unfaithful workers, we took the majority vote of the 5 labels per-pair. Human accuracy was then calculated to be 93.3%, with a precision of 98% and a recall of 89.4%.

We observed that the average response time for humans was less than 4.5 seconds/pair, with a minimum response time of half a second. This quick response average prompted us to examine mislabeled pairs: we show examples

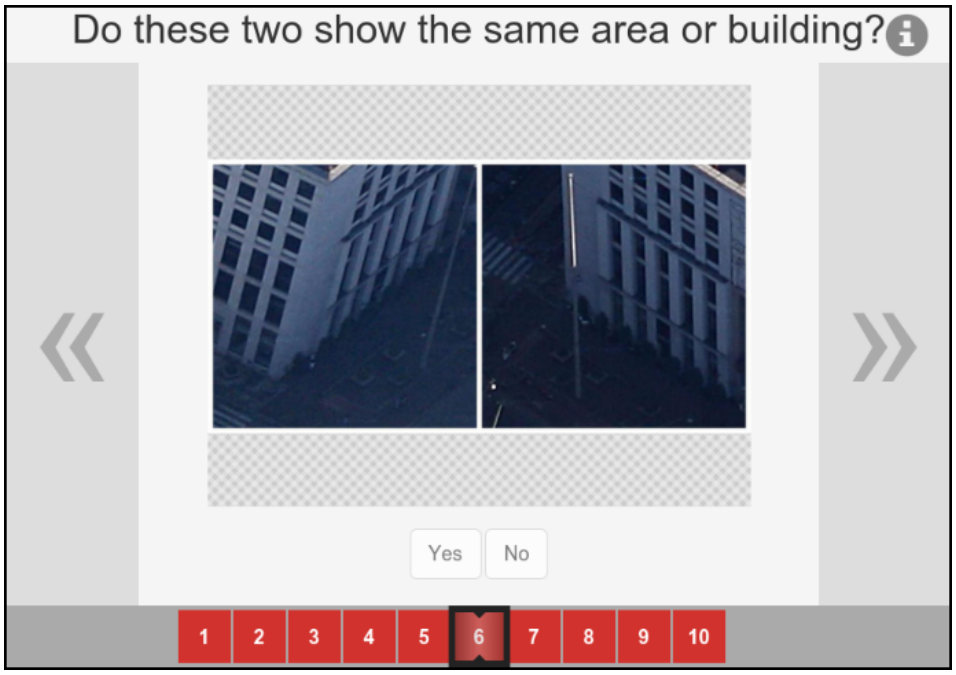


Figure 3.6: The user interface presented to our human subjects through Amazon Mechanical Turk.

of False-Positives in Fig. 3.7 and False-Negatives in Fig. 3.8. Most of the False-Positive pairs have a similar general structure, a cue that humans relied on hastily—notice that these examples require deliberate correspondence matching. This is a non-trivial, time-consuming task, which explains why the human subjects, who operate in an environment favoring lower response times, labeled them as False. This is also corroborated by the high precision and lower recall of the human labelers, which is another indication that humans are performing high-level image comparisons. All in all, we believe this indicates that the human participants were relying mostly on global appearance cues, which indicates the need for local correspondence matching.

3.3.3 Training Framework

We train our networks with *Torch7* [21]. We transplant weights in our models from the pre-trained reference model *CaffeNet* available from *Caffe* [35]. For the convolutional feature arms, we keep the AlexNet layers up to 'pool5' and discard the rest. The fully connected layers of our classifier component are trained from scratch. For the patch descriptor network, *i.e.*, MatchNet [29], we transplant the 'feature'-network and the 'classification'-network as-is and freeze the learning for both.

We use Rectified Linear Units (ReLU) for all our non-linearities, and train the networks with Stochastic Gradient Descent. The spatial transformer modules are trained specifically without momentum.

3.3.4 Spatial Transformer Details

The spatial transformer regresses $|\Theta| = 4n$ parameters, where n is the number of patches per image. Each 2 parameters are taken for an x-y location in the image plane in the range $[-1, 1]$. We specify a fixed-scale interpretation, where extracted patches are always 64×64, the resolution required by MatchNet.

In the Hybrid++ network, we remove the 'pool5' and 'conv5' layers provided by AlexNet from the convolutional arms and learn a new 1×1 convolutional layer with an output size of $64 \times 13 \times 13$, performing dimensionality reduction from the 384-channel output of 'conv4'. The localization network takes a $2 \times 64 \times 13 \times 13$ input from the two convolutional arms and follows up with 3 fully-connected layers as follows: $21632 \rightarrow 1024 \rightarrow 256 \rightarrow 4n$. The initialization of the last fully-connected layer is not random; as recommended in [33], we initialize it with a zero-weight matrix and a bias specifying initial locations for the patches. In our experiments, we predict $M = 6$ patches per image, initialized to

Figure 3.7: False-Positive pairs from the human experiment.

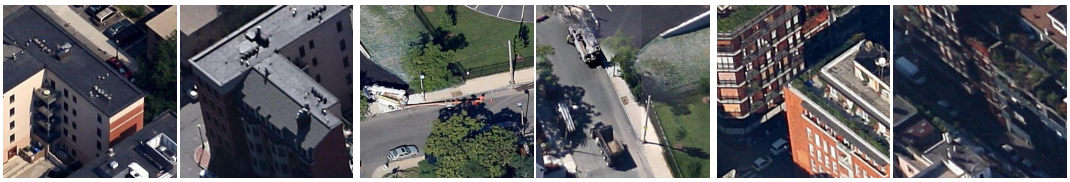


Figure 3.8: False-Negative pairs from the human experiment.

non-overlapping grid locations.

3.3.5 Matching Results

We compare our CNN models with a variety of baselines on the 'aerial' dataset. Our first baseline was a feature-based correspondence-matching method. We chose A-SIFT [80] as it offers all the capabilities of SIFT with the addition of affine invariance. In aerial images, we mainly observe affine distortion effects, which makes A-SIFT's invariance properties particularly relevant. We use the implementation offered by the authors, which computes the matches and performs outlier rejection to estimate the fundamental matrix between the views, providing a yes/no answer, given a threshold. The accuracy of A-SIFT is better than random by 11%, but suffers from low accuracy for the positive samples (*i.e.*, low recall), as it is unable to find enough correspondences to perform the fundamental matrix estimation for a large number of positive pairs. This illustrates the difficulty of this problem with local correspondence matching.

Our second set of baselines is a measure of the performance of holistic representation methods used in the image classification and retrieval literature. We

chose to compare the performance of GIST [51], Fisher Vectors [53], and VLAD [34]. The GIST-based classifier predicted most image pairs to be non-matching. Fisher Vectors surpassed A-SIFT performance by showing a better ability to recognize positive matches but performed worse than A-SIFT in distinguishing negative pairs. VLAD performed the best out of these three holistic approaches with an average accuracy of 78.6%. For GIST we use the authors' implementation, and for Fisher Vectors and VLAD we use VLFeat [71].

The third set of baselines is vanilla CNN models used in a siamese fashion (without fine-tuning). We compare against AlexNet [36], trained on ImageNet, and PlacesCNN [85], which is an instance of the AlexNet architecture trained on the Places205 dataset [85]. We extract the 'fc7' layer outputs as descriptor vectors for input images and use the L_2 distance as a similarity metric. This group of baselines explores the applicability of pre-trained networks as generic feature descriptors, for which there is mounting evidence [56]. Both CNNs performed well, considering the lack of fine-tuning. We note that while VLAD surpassed the performance of these two CNN approaches, both VLAD and Fisher Vectors require training with our dataset. This shows the power of CNNs generalizing to other domains.

Finally, we measure the classification accuracy of our proposed architectures. Our Hybrid CNN outperforms all the baselines. A variant of the Hybrid CNN was trained without the 'conv5' and 'pool5' layers, with a 1×1 convolution layer after 'conv4' to reduce the dimensionality of its output. This variant outperforms the base Hybrid CNN by a small margin. Our Hybrid++ model with Spatial Transformers gives us a further boost and performs nearly as well as the human participants in our study.

Table 3.1 summarizes the accuracy for every method, and Fig. 3.9 shows

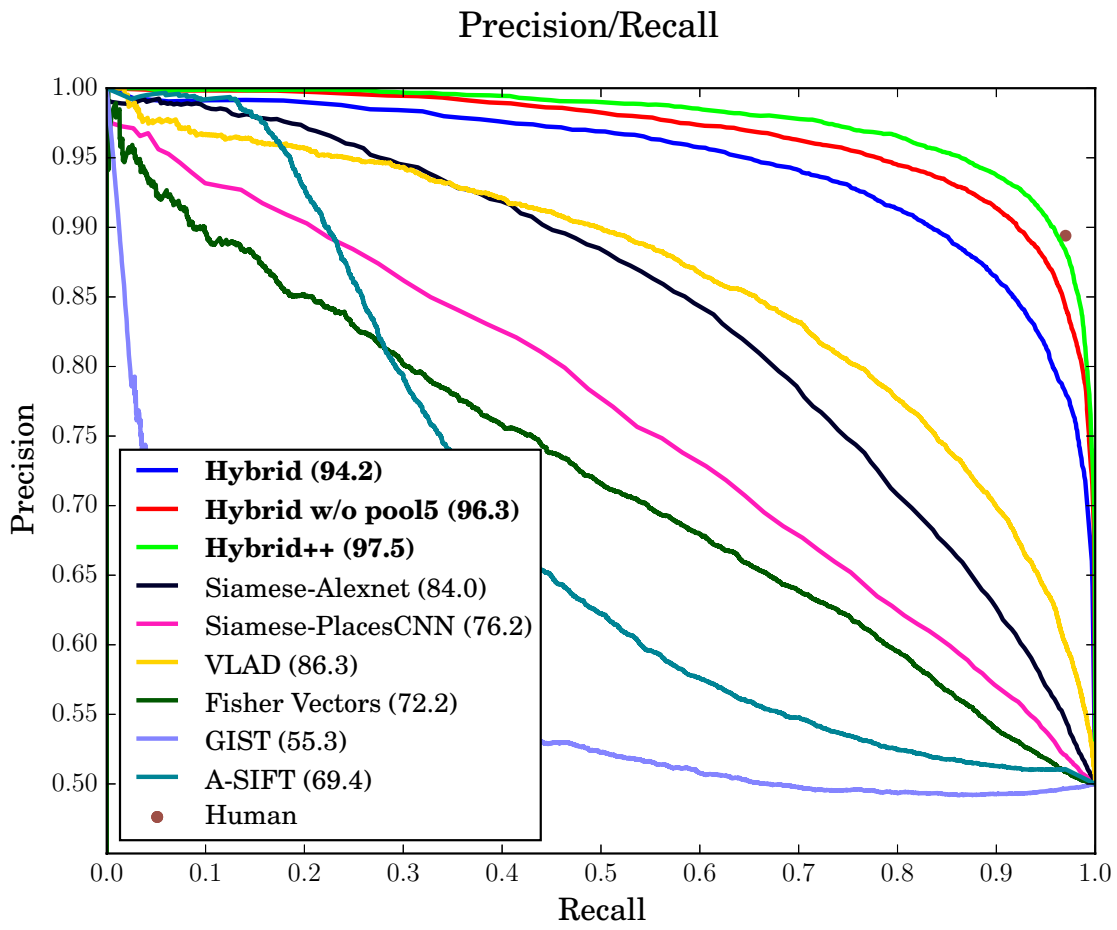


Figure 3.9: Precision/Recall curves for the 'aerial' dataset. The number between parenthesis denotes the average precision (%).

Figure 3.10: Image pairs from 'aerial', matched with Hybrid++. The overlaying boxes indicate patch proposals. Red boxes denote patches that do not match, according to MatchNet. Boxes with colors other than red indicate matches, with the color encoding the correspondence.

precision/recall curves, along with the average precision, expressed as a percentage.

Method	Acc.	Acc. pos	Acc. neg	AP
Human*	.933	.894	.972	—
A-SIFT [80]	.613	.353	.874	.694
GIST [51]	.549	.242	.821	.553
Fisher Vectors [53]	.659	.605	.713	.722
VLAD [34]	.786	.769	.803	.863
Siamese PlacesCNN [85]	.690	.626	.754	.762
Siamese AlexNet [36]	.754	.697	.811	.840
Hybrid CNN	.881	.901	.861	.942
Hybrid w/o pool5	.909	.928	.891	.963
Hybrid++	.926	.927	.925	.975

Table 3.1: Classification performance on the 'aerial' dataset. AP denotes Average Precision. (*Human performance was measured on a subset of the samples.)

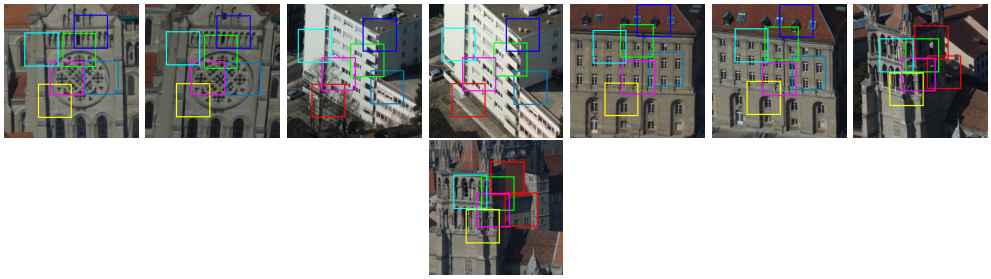


Figure 3.11: Image pairs from 'Lausanne', matched with Hybrid++. Color coding follows the same conventions are the figure above.

3.3.6 Insights and Discussion

One of the main difficulties in the application of CNNs to real-world problems lies in designing and training the networks. This is particularly true for complex architectures with multiple components, such as our Hybrid++ network. In this section, we discuss our experience and attempt to offer insights that may not be immediately obvious.

We obtained a small improvement by removing the 'pool5' layer from the AlexNet model and replacing 'conv5' by a 1×1 dimensionality reduction convolution. We believe this is mainly due to the increased resolution of 13×13 presented to the classifier. This resolution would typically allow for more local

detail to be considered jointly. In particular, this detail appears to be crucial to training the Hybrid++ model, as it provided the Spatial Transformer module with more resolution to work with. In Fig. 3.10 we show a sample of matched images with probable patch matches highlighted. Even with the increase in resolution, the receptive field for each neuron is still quite large in the original image space. This suggests that higher resolution features would be needed for finer localization of similar patches. This aspect is reflected in the network learning regions of interest for each of its attention mechanisms.

We attempted to use transformations with more degrees of freedom with the Spatial Transformer module, such as affine transforms, but we found the task increasingly difficult without higher levels of supervision and additional constraints. This was the origin of our 'out-of-bounds' penalty term. For example, the network would learn to stretch parts of each image into seemingly similar looking patches, effectively minimizing the pairwise patch similarity loss term.

To train the pairwise patch similarity portion of the network, we only have the image-level match label, with no information regarding pixel-wise correspondence. It might seem unclear what target labels should be presented to the pairwise similarity loss. However, by studying the loss function we can see that the attention mechanism would not be able to find matching patches unless we actively look for correspondences; hence it is sensible to use the image-level label for patch correspondence. Given that MatchNet modules are frozen, the network will not induce a high loss for non-corresponding patches over negative samples, but only for non-corresponding patches over positive samples.

3.3.7 Investigating the Spatial Transformers

The patch proposal locations of Fig. 3.10 are meaningful from pair to pair, and across the images for a given pair. However, while the baseline between the two images in a pair is very large, it does not change much from pair to pair—an inevitable artifact of the dataset collection process. This results in patch proposals with similar configurations and raises questions about the Spatial Transformers.

We thus set up a second experiment to study the effect of varying viewpoint changes explicitly. To this end we used several high-resolution aerial images from the city of Lausanne, Switzerland, to build a Structure-from-Motion dataset [76] and extract corresponding patches, with 8.7k training pairs and 3.6k test pairs. Patches were extracted around SIFT locations and are thus significantly easier to match than those in the 'aerial' dataset. However, the viewpoint changes from pair to pair are much more pronounced.

We followed the same methodology as before to train our models on this new dataset. In Fig. 3.11 we show different pairs from the new dataset, along with the probable patch matches suggested by the model. The model learns to predict patch locations that are consistent with the change in perspective, while also differing from pair to pair. MatchNet results on the proposals corroborate the findings when the contents of those patches do match (non-red boxes), and when they do not (red boxes). Numerical results are provided in Table 3.2. As this data is significantly easier, the baselines (notably A-SIFT) perform much better, but our method achieves the highest accuracy of 96%. The performance gain from Hybrid to Hybrid++ is however negligible.

Method	Acc.	Acc. pos	Acc. neg	AP
A-SIFT [80]	.947	.896	.998	.968
GIST [51]	.856	.798	.914	.937
Fisher Vectors [53]	.769	.723	.816	.867
VLAD [34]	.898	.867	.930	.965
Siamese PlacesCNN [85]	.690	.626	.754	.958
Siamese AlexNet [36]	.754	.697	.811	.968
Hybrid CNN	.959	.960	.957	.992
Hybrid++	.959	.962	.956	.992

Table 3.2: Classification performance on the 'Lausanne' dataset.

3.4 Conclusions

We present two neural network architectures to address the problem of ultra-wide baseline image matching. First, we fine-tune a pre-trained AlexNet model over aerial data, with a siamese architecture for feature extraction and a binary classifier. This network proves capable of discerning image-level correspondence but is agnostic to local correspondence. We then show how to integrate Spatial Transformer modules to predict probable patch matches in addition to the classification task, which further boosts performance. Our models achieve state-of-the-art accuracy in ultra-wide baseline matching and close the gap with human performance. We also demonstrate the adaptability of our approach on a new dataset with varied viewpoint changes which the ST modules can adapt to.

This work is a step towards bridging the gap between neural networks and traditional image-matching techniques based on local correspondence, in a framework that is trainable end-to-end. We intend to build on it in the following directions. First, we plan to explore means to increase the resolution of the localization network to obtain finer-grained patch proposals. Second, we plan to replace MatchNet with 'descriptor' networks trained for this specific

purpose. Third, we are interested in richer transformations for the ST modules, *e.g.*, affine, and in exploring constraints in order to do so. Finally, we want to study the use of higher supervision for a better feature-localization step, bringing neural networks closer to local correspondence techniques.

CHAPTER 4

LEARNING TO DETECT AND MATCH KEYPOINTS

The extraction of effective features is a key step in many machine learning and computer vision algorithms and their applications. In computer vision, one form of feature extraction is concerned with the detection and description of important image regions. Traditionally, these features are extracted using hand engineered detectors and descriptors. Approaches adopting this paradigm are generally referred to as *keypoint-based* or *feature-based* approaches.

Recently, the reintroduction of neural networks into many computer vision tasks broadly replaced hand-engineered feature-based approaches. Neural network based approaches generally learn the feature extraction as part of an end-to-end pipeline. While these approaches have shown great success in tasks such as scene recognition, object detection, and classification, other tasks such as structure-from-motion still depend on purely engineered features, *e.g*. SIFT [45], to detect and describe keypoints.

In this chapter, we propose a model that learns what constitutes a good keypoint, is capable of capturing keypoints at multiple scales and learns to decide whether two keypoints match. We achieve multiscale keypoint detection with a fully-convolutional network that recursively applies convolutions to regresses keypoint scores. With each successive convolution, the network evaluates image patches, i.e., keypoints, at a larger scale. By extracting the keypoint feature map after each convolution we obtain a feature map that resembles a keypoint scale-space. To learn descriptors for keypoint matching, we leverage a triplet network to learn an embedding where patches of matching keypoints are closer to each other than non-matching patches. Figure 4.1 provides an overview of our proposed model.

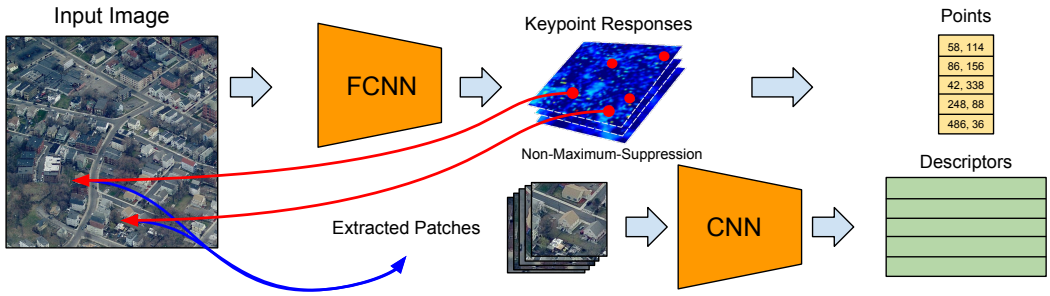


Figure 4.1: Proposed architecture for learning to detect and describe keypoints at multiple-scales. Given an image, a fully-convolutional recursive network outputs a scale-pyramid of keypoint responses, which are used to extract patches. Then, the patches are described by a patch descriptor network.

There is currently no large-scale dataset for learning both keypoint detectors and descriptors from image patches. Furthermore, finding training examples to train deep neural networks for this task poses a serious challenge, as collecting human annotated examples would be prohibitively expensive. Therefore, we create our own dataset by following a self-supervised approach, where we utilize structure-from-motion to build a large database of keypoints and matching image patches. Although those feature matches were determined originally with engineered features, structure-from-motion also factors in the underlying geometry. We only consider those keypoints that went through rigorous geometric filtering, which allows the learning of features that extend upon their engineered counterparts.

To create our supervisory examples, we collect a dataset of aerial oblique imagery and construct a large-scale model of 1.3 million 3D points using VisualSFM [76, 77]. We used those 3D points to extract matching patches exhibiting varying photometric differences, including scale, illumination, and perspective. Those patches formed the basis from which our deep neural network model was able to learn to detect and match keypoints.

We evaluate the proposed model both quantitatively and qualitatively and

show that it is capable of identifying keypoints at multiple scales as well as matching them.

Our main contributions are:

1. We propose a novel approach capable of learning to detect multiscale keypoints and descriptors for effective correspondence matching.
2. We introduce a large-scale dataset composed of over 2.5 million matching image patches at varying scales.

4.1 Related Work

4.1.1 Feature Extraction and Description

The computer vision literature has served up a large number of engineered feature extractors and descriptors, such as SIFT [45], HOG [23], SURF [9], ORB [14], and BRISK [40]. These extractors and descriptors were designed with multiple goals in mind, such as optimizing for matching accuracy or extraction and matching speeds. In general, they have been demonstrated to perform well in various applications of computer vision. Furthermore, the literature has seen approaches that learn keypoint detectors [31, 72] and descriptors [13, 62, 6]. We contrast this work by striving to learn both the keypoint detector and descriptor.

In correspondence matching problems, descriptors are used to find geometrical relationships between two or more sets of keypoints, which are then filtered by imposing geometrical constraints through model fitting techniques such as RANSAC [26]. Structure-from-motion solutions, *e.g.* VisualSFM [76, 77], start with correspondence matching and expand the computed relationships to many images building a global model governing all. In this work, we lever-

age the compounded effect of geometry on engineered features to provide our supervisory signal.

4.1.2 Deep-learning and Matching Images

In recent years, the computer vision literature has seen a surge of state-of-the-art results, on all fronts, surfacing from research on Deep Convolutional Neural Networks [36, 64, 65, 27].

In [81, 29, 61], deep architectures were proposed to learn feature descriptors. The siamese architecture [12] forms the basis for these approaches, with the neural networks learning to embed 64×64 patches in a feature space where matching patches are closer to each other than non-matching patches. Their supervisory signal is based on structure-from-motion patches originally used in [13]. However, they do not learn keypoint detection and do not handle various scales natively. We build on these approaches by showing how to create a model that learns to predict the keypoints and their respective descriptors at various scales.

The detection of salient regions with deep architectures has been mostly discussed within the object detection and recognition literature. In [44], features in later layers were shown to correspond to fine details in the receptive fields covered by those features. One approach to generate salient region proposals are visual attention models, e.g., [49, 5]. There, a recurrent network is trained to examine and propose regions of the image space sequentially. Attention mechanisms typically learn the salient features in an unsupervised manner. One particular approach is the Spatial Transformer Network [33] which describes a region proposal scheme capable of highlighting regions with associated transformations to a canonical pose that is learned automatically. In Chapter 3, spatial trans-

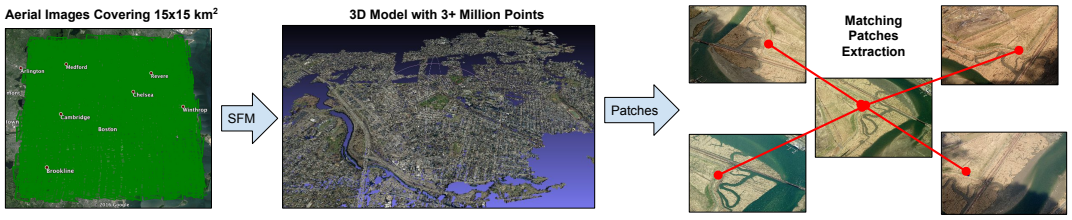


Figure 4.2: Generating multiscale matching patches using structure-from-motion.

former networks are used to detect a fixed number of probable patch matches. In essence, there the network attempts to detect and match patches simultaneously, with only weak-supervision from match/no-match labels on the image level. Another approach to generating region proposals are Region Proposal Networks [27] that have the sole purpose of identifying regions of the image space that contain objects. Region proposal networks are generally trained in a fully supervised manner. This approach has been recently extended [57] by coupling the proposal network with the classification network for faster performance. We draw inspiration from these works for modeling a network capable of proposing keypoints.

4.2 Learning Model

The goal of this model is to learn to detect and match keypoints in images. We achieve this by using two models, one for each task. The first is a keypoint detection network, and the second is a keypoint description network.

4.2.1 Training Data

To train the keypoint detection network and the keypoint matching network, we require a large set of patches with high-quality keypoints that are also annotated

with pairwise match information. Since no such large scale dataset currently exists, a key aspect of our data-driven learning approach is the collection of a large-scale training dataset. For that purpose, we follow a self-supervised data collection scheme.

Our data generation approach is similar to that of [13]. We rely on structure-from-motion techniques to identify good keypoints and to generate matching patches. However, our approach differs in that we keep the original patch sizes, without rescaling to a canonical size. This allows us to train a multiscale detector.

We use aerial imagery covering an area of $15 \times 15km^2$ around the city of Boston, Massachusetts to construct a 3D model using VisualSFM [76, 77]. The model contains 1.3 million 3D points where each is observed from at least two cameras, *i.e.* images. For a single 3D point with k associated keypoints, there are $k(k-1)/2$ unique keypoint pairs that we can extract as matching patch pairs. To generate patches that do not constitute good keypoints, we randomly sample image patches that do not belong to any keypoints. Figure 4.2 gives an overview of the approach.

The keypoint scales extracted from the 3D model are continuously valued. For our approach, we discretize the scale values into five scales: $S = \{64, 96, 128, 192, 256\}$. We determined the set of scales by clustering the scale ranges in the extracted dataset. As we show later, this discretization does not limit the model. The fixed scale range directly affects our design, however only in one way: the smallest scale the model handles is 64×64. There is, however, no limit on the largest scale. This is a result of the recursive architecture, which we will discuss in the following subsection.

We denote the generated set of patches P as follows:

$$P = \{p_i : (x_i, s_i, k_i); \quad k \in \{-1, 1\}\} \tag{4.1}$$

where p_i is a patch with: x_i as the raw pixels, s_i is the scale of the patch, and k_i is the keypoint label. Further, we denote the generated set of matches M as

$$M = \{m_i : (p_j, p_k, y_i); \quad p_{j,k} \in P, \quad y_i \in \{-1, 1\}\} \tag{4.2}$$

where each match m_i is tuple that references two patches p_j and p_k with y_i being the match true/false label.

4.2.2 Detection Network

The goal of the detection network is to identify the regions of the input image that constitute good keypoints. Identifying keypoint regions includes both finding the optimal keypoint locations as well as their scales. In particular, we learn a nonlinear function $f(X)$, from images X into a feature space $\mathbb{R}^{w \times h}$, where high activations correspond to respective image regions that constitute good keypoints. The architecture used for training the detection network differs slightly from the architecture used during inference since it is trained on image patches, but the inference is performed on whole images.

Training Procedure

Figure 4.3 illustrates the training architecture. The input to the network are batches of patches $\{p_i\} \subset P$ and associated binary labels indicating whether the patches represent good keypoints. In essence, the detection network is a binary classification CNN that learns to decide whether a given patch constitutes a good keypoint or not. As such, it consists of a sequence of convolutional and pooling layers followed by two fully-connected layers for classification.

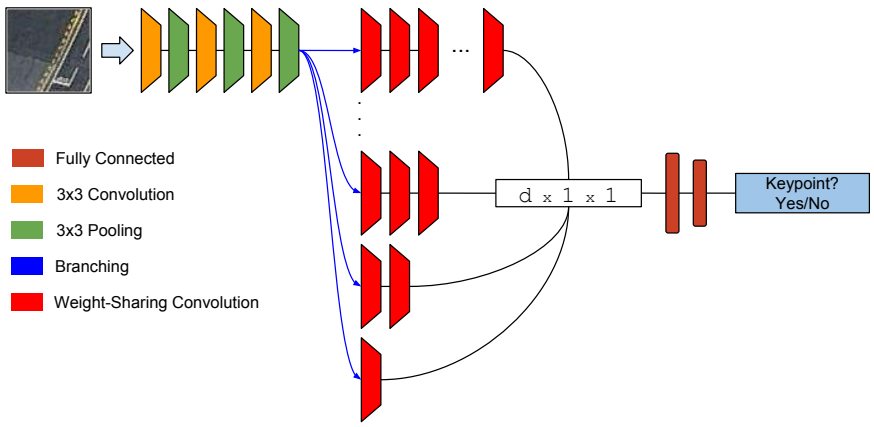


Figure 4.3: Training architecture for multiscale keypoint detection network. First, patches pass through a set of convolutions and pooling layers. Then, a recursive convolution is applied until the feature map dimension is 1×1. Since a batch can contain patches of different scales, a scale-dependent branch is chosen for each patch determining the number of recursive convolutions. Finally, two fully connected layers lead into a binary keypoint classifier.

Keypoints vary largely with respect to their scale and thus the patches come in many different sizes. To address this, we propose a scale-dependent branching mechanism, shown in Figure 4.3 by blue arrows. There, a scale-dependent branch is chosen for each patch. Within each branch, convolutional filters are applied recursively until the output feature is of dimension $(d \times 1 \times 1)$. This allows for encoding keypoints of varying scales in a common feature space of fixed size. This is essential for efficient multiscale inference. All convolutions across all scale-dependent branches share the same weights, allowing for inference over arbitrarily large input images. In essence, the recursive application of the same convolutional filters resembles a rolled-out recurrent neural network for handling multiscale inputs. The d-dimensional output from the recursive branches is then used to determine whether the patch is centered around a good keypoint.

To sample training patches, we use hard-negative mining to improve the performance of the keypoint detector. For each training batch, we randomly

sample the dataset searching for patches with high-loss, to construct batches of difficult examples. Each batch is chosen to have a mix of positives and negatives with a 1:1 ratio.

Training Objective

We define a loss function $\mathcal{L}_{\text{KP}}$ for training the keypoint detection network. The loss function comprises two terms. First, as we model keypoint detection as a binary classification problem, we make use of the hinge-loss to define our first term. Second, we use a squared difference loss to penalize network responses on non-centered patches. This employs a Gaussian-like response around the center of the patch and is inspired by the response shape penalty used in [72]:

$$h_j = e^{-\frac{\left\|v_j\right\|_2}{2\sigma^2}} \tag{4.3}$$

where v_j is the vector from the keypoint towards the center of the patch. During training, non-centered patches are generated by extracting patches jittered around the keypoints. This results in data-augmentation as well as incentivizes maximum responses around the centers of informative regions. Putting the two terms together, the joint loss function is given as

$$\mathcal{L}_{\text{KP}} = \frac{1}{N} \sum_j \left[\lambda \max\Big(0, 1 - y_j x_j\Big) + (1-\lambda)\Big(x_j - h_j\Big)^2 \right] \tag{4.4}$$

with x_j as network output, $y_j \in \{-1, 1\}$ as the training label, and λ as mixing-weight.

Inference

Figure 4.4 depicts the inference architecture, which differs slightly from the training architecture. During inference, the network processes whole images,

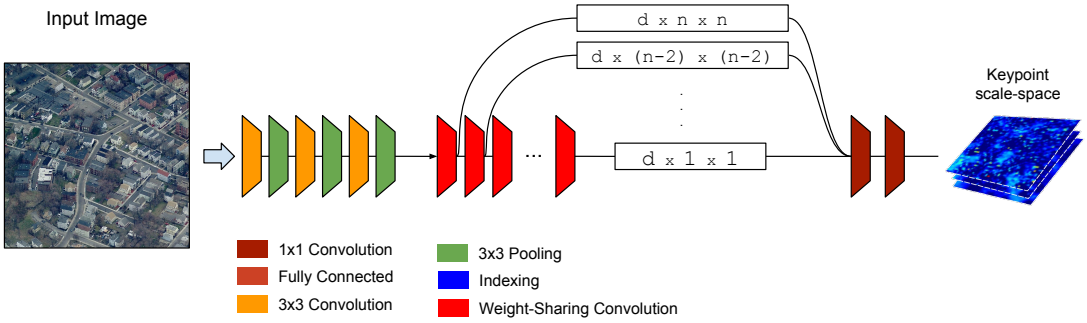


Figure 4.4: Inference architecture for multiscale keypoint detection network. First, an input image is passed through a set of convolutions and pooling layers. Then, a recursive convolution is applied until the feature map dimension is 1×1. After each recursive convolution we compute the keypoint feature map. Since convolutions later in the network have larger receptive fields the output feature maps resemble a keypoint *scale-space*.

as opposed to patch-sized inputs. However, we assume that input images are at least of size 64×64.

Instead of a single value describing the keypoint quality of a single patch, the network is converted to be fully convolutional as to output a feature map. There, each value corresponds to the keypoint score of a specific image region. In particular, we compute the keypoint feature map after each recursive convolution. As inputs progress deeper into the network, the receptive field of individual neurons increases so that larger patches in the input image are considered. As a result, the output feature maps resemble a keypoint *scale-space*. We illustrate this in Figure 4.5. This allows us to select the best scale for each patch by finding the scale with the highest keypoint response score. Finally, the best keypoints are extracted with non-maximum suppression.

4.2.3 Description Network

The goal of the descriptor network is to learn a nonlinear feature embedding $f(p)$, from patches p into a feature space $\mathbb{R}^d$, such that for a pair of patches p_1 and p_2, the Euclidean distance between $f(p_1)$ and $f(p_2)$ is small if the patches match and is large if they do not match. The training follows an approach similar to the triplet network proposed in [59]. In particular, the nonlinear embedding should ensure that a patch p_1 (anchor) is closer to all patches depicting the same keypoint p_2 (positive) than it is to any other patch p_3 (negative). Given the feature embedding and sets of keypoints with respective patches, the best matching keypoint can be found by retrieving nearest neighbors in the embedding space.

Training Procedure

Figure 4.6 illustrates the training architecture. The input to the network are batches of patch triplets $\{p_1, p_2, p_3\}$. First, each patch is fed through a convolutional neural network to compute its embedding feature vector, where three networks share the same weights. The feature vectors are then normalized to lie on the d-dimensional unit hypersphere. Afterward, the pairwise Euclidean distances between the feature vectors of the anchor and the two other patches are computed. The network is then supervised with a triplet ranking loss shown in Equation 4.5 to project the matching patches closer in the feature space and the non-matching patches.

The patch triplets are sampled online. The anchor and the positive match are drawn from the match set M. The negative patches are sampled at random. In order to ensure good convergence, it is important to sample triplets that induce loss, i.e., they violate the triplet constraint. To get triplets that violate the constraints, we perform online hard negative mining. In particular, for each

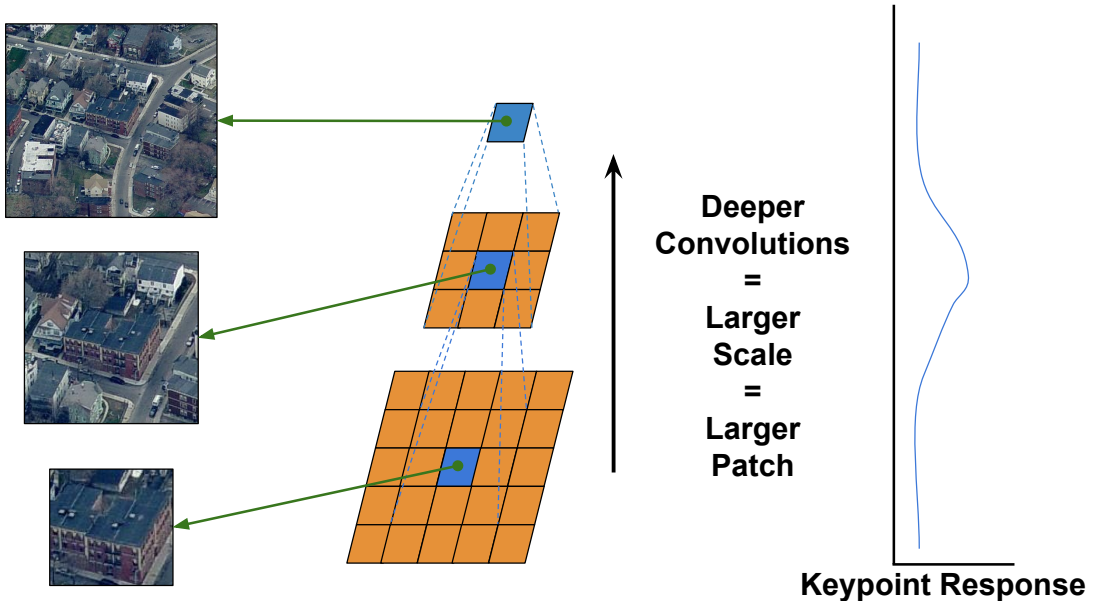


Figure 4.5: Convolutions later in the detection network correspond to larger patch sizes. The keypoint feature map with the highest response indicates the best keypoint scale.

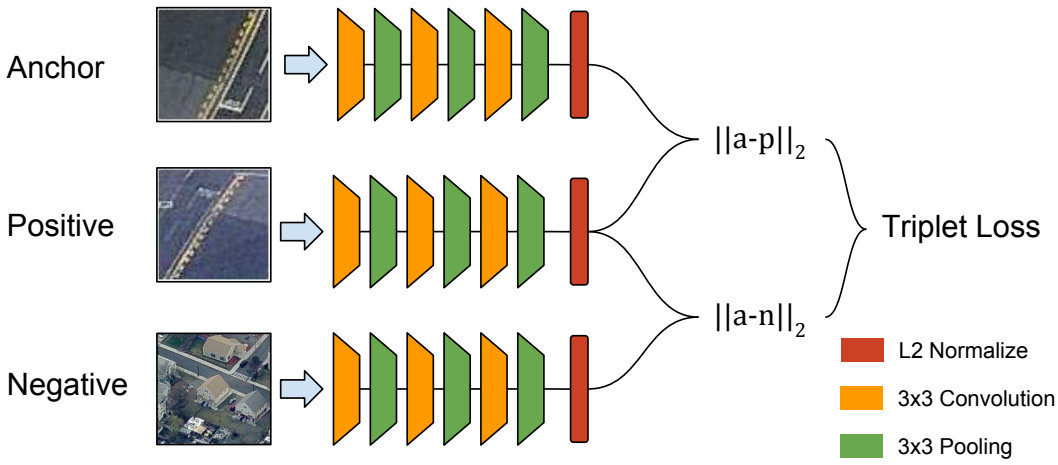


Figure 4.6: Training architecture of the keypoint description triplet network. Three patches are passed through channels which share weights to rank their euclidean distances in the feature space.

matching pair (anchor and positive) in the training batch, we choose a negative patch in the batch that violates the triplet constraint the most. All matching pairs within a batch can choose from the same set of negative patches.

Training Objective

We define a loss-function $\mathcal{L}_{\mathrm{T}}$ for training the keypoint description network, as follows. Given triplets $T = \{t_j : (p_1^j, p_2^j, p_3^j)\}$ and a scalar margin h, the loss function is given by:

$$\mathcal{L}_{\mathrm{T}} = \frac{1}{N} \sum_j \left[\max \left(0,\ D(p_1^j, p_2^j) - D(p_1^j, p_3^j) + h \right) \right] \tag{4.5}$$

where h is chosen as 0.2 and D is a Euclidean distance function defined on the embedding feature vectors, which are computed from the image patches.

$$D(p_a, p_b) = \| f(p_a) - f(p_b) \|_2 \tag{4.6}$$

4.2.4 Full Model

After both networks are trained, keypoint detection and matching can be performed. The process is similar to the traditional keypoint extraction and description pipeline.

First, a whole image is fed through the fully convolutional detection network. A sample output is shown in Figure 4.8. From the output feature map, a set of keypoints are extracted by filtering with non-maximum suppression. Then, for each keypoint, we crop a patch according to the detected scale and rescale it to 64×64, the canonical patch size of the description network. Subsequently, the keypoint descriptors are computed with the triplet network. Finally, given the keypoint descriptions for two images, corresponding keypoints are found using nearest neighbor search.

Model Component	Structure
Feature Detection	C3/128/2-B-P3/2-C3/128/1-B-P3/2-C3/d/1-R{C3/d/1-B}
Keypoint Scoring	C3/64/1-B-C1/1/1
Patch Matching	C3/128/2-B-P3/2-C3/256/1-B-P3/2-C3/256/1-B-P3/2-L2N

Table 4.1: Network structure parameters. *Convolution* is denoted with **C**$k/f/s$, where k is the kernel size, f is the number of filters or outputs, and s is the stride. Similarly, *Max Pooling* is denoted with **P**k/s, batch normalization is **B**, and fully-connected layers are **FC**. R stands for "repeat", and $L2N$ is for $L2$ normalization. Parameter d denotes the number of filters, in the convolutional layers that vary among experiments.

4.3 Experiments

4.3.1 Experimental Setup and Model Parameters

We verify the effectiveness of our model by testing on a separate held-out test set, which was formed by removing cameras (images) from the structure-from-motion 3D model prior to training. The held-out set is comprised of about 800K patches of varying scale, with matching information.

The specific parameters of the networks used in the experiments are described in Table 4.1. All convolutions are without padding. For optimization, we used Stochastic Gradient Descent with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.005.

4.3.2 Keypoint Detection

To test the keypoint detector, we run different versions of the keypoint-detection network, and compute precision/recall for each. The networks differ in the number of feature dimensions (referred to with d in Table 4.1), and hard-negative mining procedure.

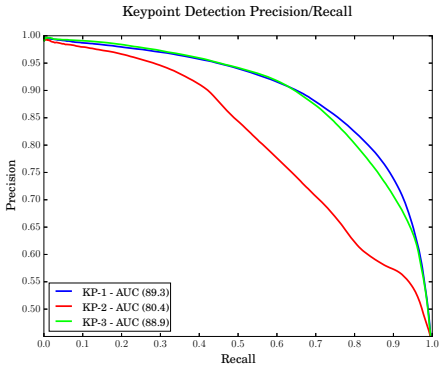


Figure 4.7: Precision/Recall curves for different variants of our keypoint detector.

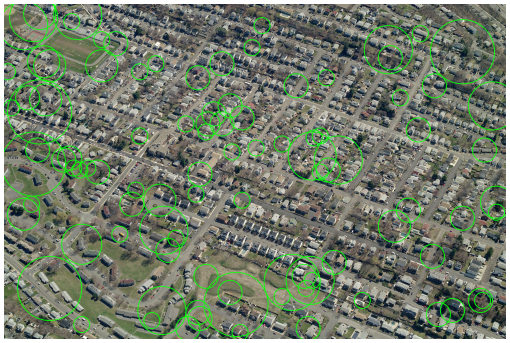


Figure 4.8: Sample keypoint detections on a full-sized image.

Our first network "KP-1" has $d = 256$ and uses hard-negative mining from the first iteration. The second network "KP-2" has $d = 256$ and uses hard-negative mining starting from mid-training with the whole batch comprised of hard-negatives. The last network "KP-3" has $d = 128$ and follows the same hard-negative mining procedure as "KP-1". The precision/recall curves are shown in Figure 4.7.

The results indicate that using hard-negative mining from early training allows the model to find a better solution as opposed to start using hard-negatives only during mid-training. One explanation could be that the model may have already arrived at a good local minimum for identifying keypoints. The results also show added benefits from additional model parameters. Overall, the model performance well with an area-under-the precision-recall curve of 89.3 on keypoints of varying scales.

4.3.3 Patch Matching

To evaluate our triplet-based patch matching network, we compare against DeepCompare [81] and MatchNet [29], which both leverage a siamese-based

architecture. The evaluation is based on a retrieval framework. For a randomly sampled pair of matching patches in the test set, we use one of the patches as *probe* and the other as *target*. The target is mixed with a set of nonmatching patches, for a total set of 100 patches. Then, given the probe, the task is to find the matching patch within that set.

In our evaluation, we report the retrieval at rank 1 (top-1%) and within ranks [1-5] (top-5%). The test includes two variants of our network. The first variant "Triplet-1" is based on a small number of convolution and pooling layers such as shown in Figure 4.6. The second variant "Triplet-2" is based on the VGG-16 network [16]. We run each network and rank the matches according to distance or similarity (MatchNet and DeepCompare, both follow a similarity metric). Our test set contains 5K matching pairs, randomly sampled from the test set. We perform two runs independently and report the average in Table 4.2. For DeepCompare [81], we report results only for the two best variants (out of five) for brevity. The results show that the proposed method outperforms previous results. We believe this is due to the structure of the embedding learned by the triplet loss function which is more suitable for ranking purposes.

4.3.4 Extending to Other Datasets

To evaluate the generalization of the learned keypoint detector and descriptor, we present qualitative results for our learned models on a dataset with different image statistics. In particular, we applied our models on the "Wall" sequence from the Oxford dataset [48].

In Figure 4.9, we show the five-image sequence comparing the first image with the rest of the images in the sequence. Our network shows good results in the first two images, retrieving the correct homography. The results in the third

Figure 4.9: Qualitative evaluation of feature transferability: Keypoint detection and matching results from network trained on aerial imagery and tested on "Wall' image from the Oxford dataset [48]. For the first two images, the network successfully retrieves the correct homography. The result on the third is partly correct. The last two image demonstrate failure cases.

image are partly correct and the last two image demonstrate failure cases. The type of images differs largely between our training dataset and the test images. However, the approach shows promising results indicating good capabilities of extending to other datasets.

4.4 Conclusion

Feature extraction and description is a central problem in computer vision. We presented a novel deep learning architecture capable of multiscale keypoint detection and description. Our approach serves as a step to bring classical approaches closer together with the recent progress in deep learning. We plan to further investigate the model's performance on other benchmarks and explore other avenues for multiscale detection and description.

Method	Top-1%	Top-5%
Triplet-1	**73.8**	**93.4**
Triplet-2	**76.6**	**95.5**
MatchNet [29] - Liberty	57.3	82.3
MatchNet - Yosemite	44.0	73.1
MatchNet - Notredame	52.5	78.6
DeepCompare [81]- 2ch - Liberty	71.1	88.7
DeepCompare - 2ch - Yosemite	70.9	88.6
DeepCompare - 2ch - Notredame	71.9	88.0
DeepCompare - siam - Liberty	67.6	90.0
DeepCompare - siam - Yosemite	70	88.6
DeepCompare - siam - Notredame	70.7	91.0

Table 4.2: Retrieval at rank 1 (top-1%) and within ranks [1-5] (top-5%) on our test-set.

CHAPTER 5

LEARNING FROM HALLUCINATIONS

In computer vision applications, two varying approaches had been historically adopted to solve problems; namely, feature-based and direct methods. Nowadays, these earlier distinctions no longer hold exclusively, however, they heavily influence modern approaches. The initial motive for feature-based approaches had been the reduction of computation requirements for vision algorithms. Research in feature-based approaches led to the development of many algorithms for two separate tasks. The first task is to *detect* and identify salient regions within an image, for the purpose of only processing those salient regions while discarding the remainder of the pixels. The second task is to *describe* the contents of those salient regions. Computer vision algorithms would proceed by performing these two tasks and using the computed features to accomplish more complex goals, such as finding correspondences between two images, or detecting a certain object within an image, for example.

We concern ourselves in this chapter with feature description embodied as patch descriptors. In recent years, approaches to feature description have leveraged discriminative learning methods to learn descriptors superior to manually engineered variants. The learning approaches have been championed by neural networks and deep learning achieving state-of-the-art performance. One of the main criticisms of neural-based approaches to patch description is the requirement of immensely large numbers of matching patches to learn proper descriptors that generalize well. The requirement for matching patches arises from the underlying models that aim to learn feature embedding spaces where matching patches are *closer* to each other as compared to non-matching patches. Existing methods [79, 3, 13] have mainly relied on structure-from-motion to create sparse

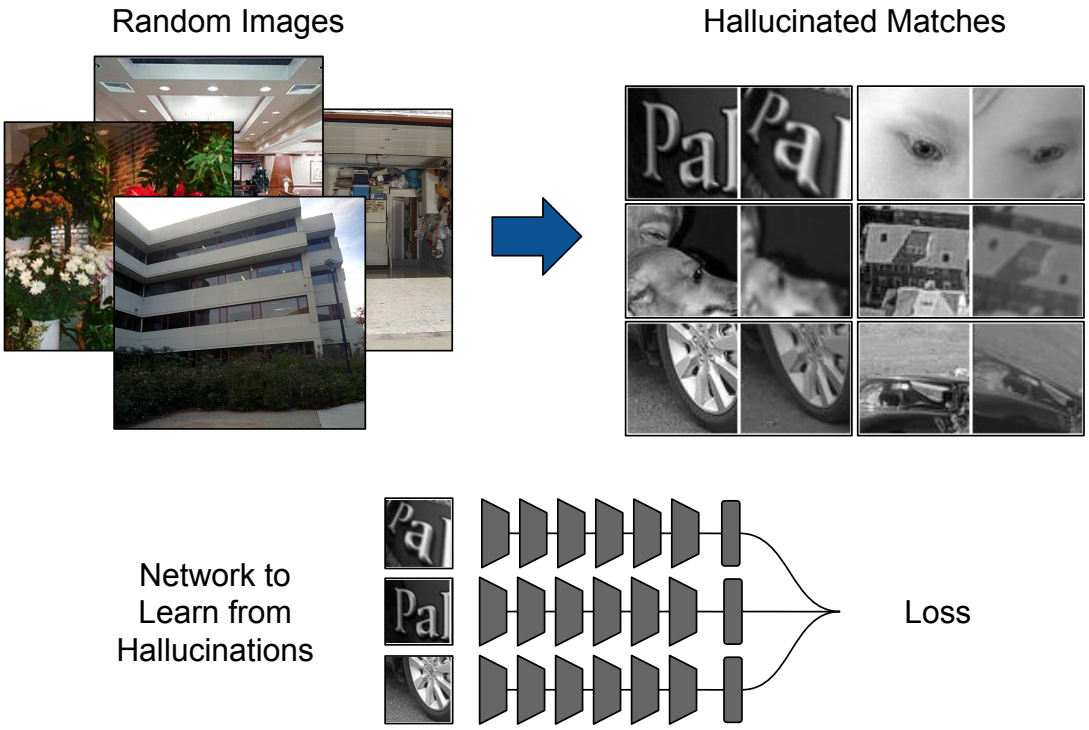


Figure 5.1: From hallucinations to patch descriptors.

3D models with many views of a single 3D point, thereby generating very large numbers of matching patches. However, engineered approaches lie at the heart of obtaining these training examples, and it is unclear whether such an approach is a limiting factor. In contrast, the work of Zamir *et al*. [82] finds training examples by establishing patch matches through combining 3D models of cities with registered images to identify high quality matches exhibiting camera rotations exceeding $120°$. The performance of their descriptor exceeded those descriptors learned with structure-from-motion, which could be attributed to the high-quality examples of many varying viewpoints.

In this chapter, we ask the question: Do we really need a large number of carefully labeled patch matches to achieve good performance? The implications of answering this question are significant, as collecting high-quality labels typically costs a large sum of money and time. If good performance can be

achieved in an unsupervised or self-supervised manner, then financial and time costs could be minimized as we would only collect high-quality labels only to bridge the gap between self-supervised performance and fully-supervised approaches.

The main difficulty in feature detection and description is achieving invariance to viewpoint and illumination changes. Approaches in the literature have sought to solve this problem through two main methods. First, *normalization*, where the shape of detected salient regions captures some local geometry resulting in the extraction of patches that appear to be in a canonical view, *e.g.* normalizing keypoint rotation in SIFT [45]. Second, *synthesis*, where descriptors are computed from many synthesized viewpoint changes, as in Affine-SIFT (A-SIFT) [80], and then used collectively.

We are here inspired by the *synthesis* camp, except that our synthesis step is only used for training and learning, *i.e.*, unlike A-SIFT where this is performed during run-time. We propose an approach where a patch descriptor is learned from vast numbers of hallucinated patch matches which are generated in an online manner. The generated examples are used to train a deep neural network to match these patches. In this scenario, the synthesized changes serve only in the learning of a feature-embedding that captures how a patch might look like from a different vantage point. Figure 5.1 gives an illustration of our proposed approach. Our evaluation then shows that the network is capable of learning a good feature representation solely from these hallucinated examples.

The contributions of this work are as follows:

- We show how ordinary images can be used to generate numerous examples for the purpose of training a patch descriptor.
- We demonstrate how hallucinations can build a model that performs very

well despite the lack of labels.

5.1 Related Work

5.1.1 Features and Geometry

Feature description and extraction have amounted to many works in the Computer Vision literature. In the early 2000s, SIFT [45] unleashed a revolution in feature-based methods. It inspired a whole set of other descriptors, *e.g*. HOG [23], ORB [14], SURF [9], and BRISK [40]. Some descriptors aimed for fast matching speeds where others optimized for quality matches.

The extraction of meaningful geometry is a paramount goal of feature-based methods. Finding correspondences by matching keypoint descriptors across pairs of images is the main ingredient in almost all geometric pipelines. Typically, matching keypoints are filtered through geometric model fitting techniques such as RANSAC [26] and its derivatives. Structure-from-motion applications, such as *e.g*. VisualSFM [76, 77], perform feature matching across many images simultaneously and build a 3D model relating all input images.

5.1.2 Discriminative Learning of Descriptors

Learning approaches to feature detection and patch description have been visited at multiple occasions [13, 62, 6, 31, 72]. With the rise of Deep Convolutional Networks in the previous years, feature detection and descriptor matching were not spared. For example, [81, 29, 61, 3, 79, 82] propose multiple approaches to learning feature detectors and patch descriptors. In [81, 29, 61] the siamese architecture of [12] is adopted with the goal of learning a feature space such that

visually similar patches are rendered closer in that space as opposed to non-visually similar patches. Approaches [3, 7] adopt similar architectures with the difference of using triplets instead of binary examples. Training labels for these approaches are mainly obtained through structure-from-motion 3D models, as in [13]. Zamir *et al*. [82] follows a different approach, where vast amounts of Google Street View data are joined with external 3D models to establish high-quality correspondences between widely separated views. The quality of their data and training approach yielded a superior descriptor to previous structure-from-motion seeded descriptors. In [18], a fully convolutional metric learning approach is used for both semantic and geometric correspondence estimation.

In a similar spirit to this work, we find [2, 43] learning to match images and patches from sequences, where the supervisory signal is solely based on the small changes found between consecutive frames. We, in this chapter, take a different approach and investigate the applicability of data generation and synthesis in this problem. We recognize that obtaining high quality labeled data is of paramount importance, however, if good performance can be obtained without high financial and time expenditure, then we will find ourselves in a win-win situation where we only need to expand in bridging the gap.

Visual attention models and their derivatives, *e.g*. [49, 5], embody instances of feature detectors and descriptors that operate in a sequential manner. Typically a recurrent neural network is trained to focus on what it deems as salient regions of the image for the purpose of classification [49, 5] or visual description [78]. Matching images with attention mechanisms was also explored in Chapter 3, where the Spatial Transformer Network [33] forms the basis for identifying salient regions for matching with [29], all within one neural architecture. An important aspect of attention approaches is the unsupervised identification of

salient regions. Here, we assume that salient regions have been identified, and we are only concerned with learning how to match salient regions by means of data generation and synthesis.

5.2 Learning from Hallucinations

5.2.1 Hallucinating: How-to

The generation of matching patches is a crucial step in learning our patch descriptor pipeline. Our approach does not hallucinate patches from scratch and is inspired by that of DeTone *et al*. [24] used to learn a homography estimator with a deep network. In our approach, we aim to identify a good patch for viewpoint change synthesis by treating the contents of the patch as if they were from a planar scene.

Given an image I, we run a feature detector to identify a set of keypoints $K = \{k_i; k_i = (x_i, y_i), x_i \in [0, W), y_i \in [0, H)\}$, where W and H are the width and height of I, respectively. Afterwards, we randomly pick a keypoint $k \in K$, and determine a patch P of size $S \times S$ with k defining its center. We define P as a tuple: $P = (p_{tl}, p_{tr}, p_{bl}, p_{br})$ of the four corners of the patch, where $p_* = (x, y)$ are pixel-coordinates. Given a maximum perturbation value V_{pix}, we randomly perturb the locations of each corner $o_* = p_* + \epsilon$, where $\epsilon \sim \text{Uniform}(-V_{pix}, V_{pix})^2$. These perturbations allow us to imagine viewing the patch from a different view-point. Moreover, we also simulate rotations, correlations between the ϵ samples. Given V_{rot}, we randomly pick a rotation angle $\theta \sim \text{Uniform}(-V_{rot}, V_{rot})$. The rotation angle θ is used to define a rotation matrix $R \in \mathcal{SO}(2)$. Afterwards, we arrive at the final perturbed pixel locations $q_* = Ro_*$.

Given the final perturbed locations $Q = (q_{tl}, q_{tr}, q_{bl}, q_{br})$, we solve for the homography H^{QP} relating the two point sets P, Q using the direct linear transformation (DLT) algorithm. We then use H^{QP} to transform I and follow that by cropping the patches defined by P from both I and its transformed version. The resulting two patches are considered *matches*. We then further process the match by simulating illumination changes through random brightness and contrast shifts. Finally, we simulate zoom effects by randomly blurring one of the patches with a gaussian filter of random size and standard deviation, both selected from $V_{blur-k} = \{3, 5, 7, 9\}$ and $V_{blur-\sigma} = \{1.5, 2, 2.5, 3, 3.5\}$. Figure 5.2 gives an overview of the generation method, and Algorithm 3 formally defines the procedure.

Algorithm 3 Algorithm to hallucinate a single matching pair. The variables are as defined in the section text.

Require: Image I, S, V_{pix}, V_{rot}, V_{blur-k}, $V_{blur-\sigma}$
  Keypoints = $DetectKeypoints(I)$
  Keypoints = $FilterNearBorder(Keypoints, I, S)$
  KpIdx = $Uniform(0, sizeof(KeyPoints))$
  pt = Keypoints[KpIdx]
  $s = S/2$
  $P = \{(pt.x-s, pt.y-s), (pt.x-s, pt.y+s), (pt.x+s, pt.y+s), (pt.x+s, pt.y-s)\}$
  $Q = copy(P)$
  for $p \in Q$ **do**
    $p = p + Uniform(-V_{pix}, V_{pix})^2$
  end for
  $\theta = Uniform(-V_{rot}, V_{rot})$
  $R = RotationMatrix(\theta)$
  for $p \in Q$ **do**
    $p = R(p - pt) + pt$
  end for
  $H = DirectLinearTransform(Q, P)$
  $I_2 = Warp(I, H)$
  $P_1 = I[P]$
  $P_2 = I_2[P]$ **return** Matching patch pair P_1, P_2

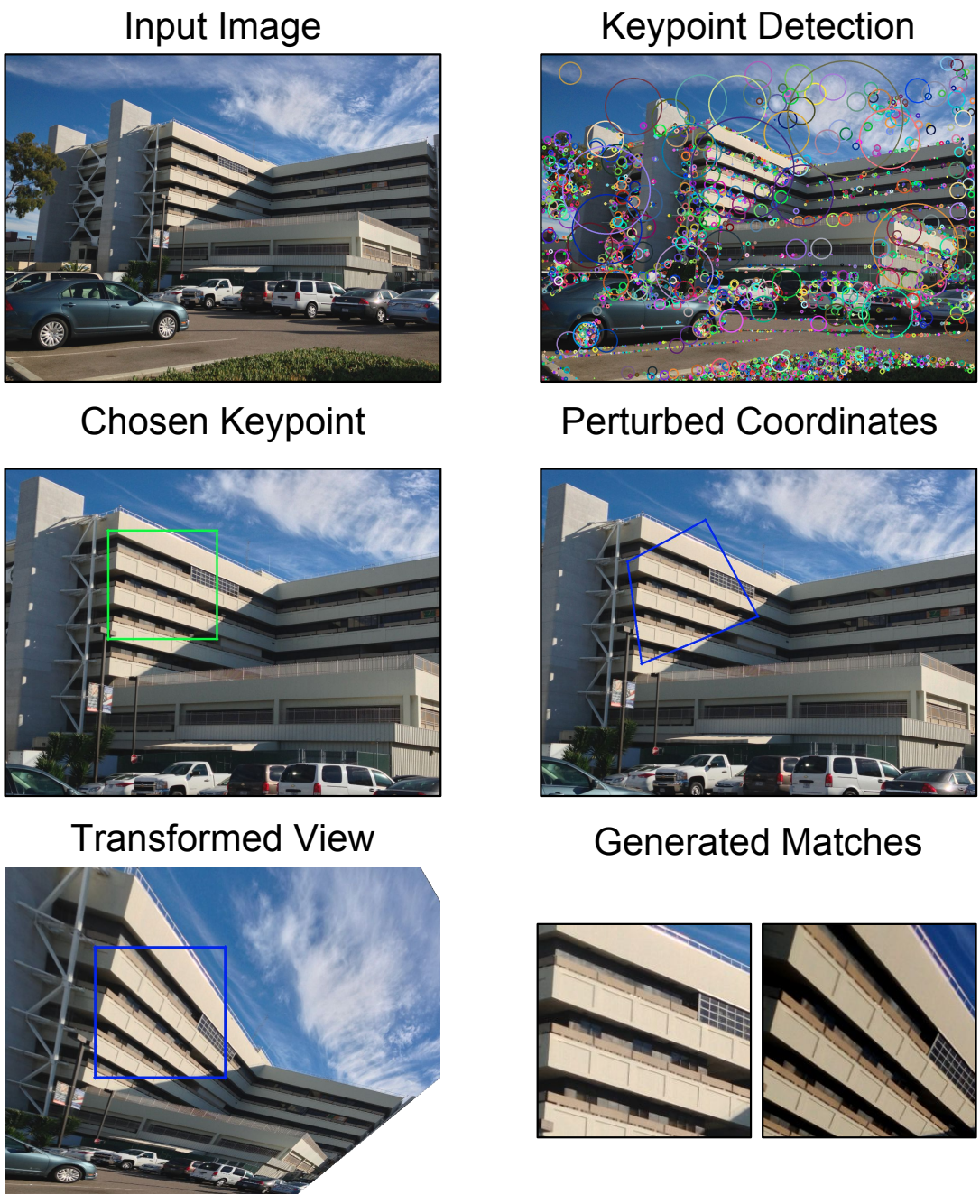


Figure 5.2: The process of hallucinating matching patches from images.

5.2.2 Learning Model

Given the method for generating synthesized patch matches. We build a neural model to learn a feature embedding for those synthesized matches. Our model is quite similar to those in the literature [79, 82, 3, 7], albeit with minor architectural differences. The main learning architecture outline is designed around a triplet loss, with three feature extraction towers that share weights. Each tower computes a feature descriptor given a color image. The loss function then drives similar patches closer to each other in the embedding space.

Specifically, we use the triplet loss function from [65] to learn our feature descriptor embedding. Given our hallucination method we can define patch triplets $T = \{t_i : (a^i, p^i, n^i)\}$, such that (a^i, p^i) are generated matching patches, and n^i is a randomly extracted patch. Further, given a scalar margin α, we define the loss function $\mathcal{L}_T$ as:

$$\mathcal{L}_{\mathrm{T}} = \frac{1}{N} \sum_{i} \left[\max \left(0,\ D(a^i, p^i) + \alpha - D(a^i, n^i) \right) \right] \tag{5.1}$$

where α is chosen as 0.2 and D is a Euclidean distance function defined on the embedding feature vectors, which are computed from the image patches.

$$D(a, b) = \| f(a) - f(b) \|_2 \tag{5.2}$$

The network architecture is shown in Figure 5.3.

5.2.3 Curricular Training

To train our model, we specifically follow an unconventional training procedure, where we attempt to get the network to learn to match easy patches first before moving on to harder examples. In essence, the method is similar to a school curriculum where easy subjects are introduced before more advanced topics, while only allowing students to advance after passing knowledge tests.

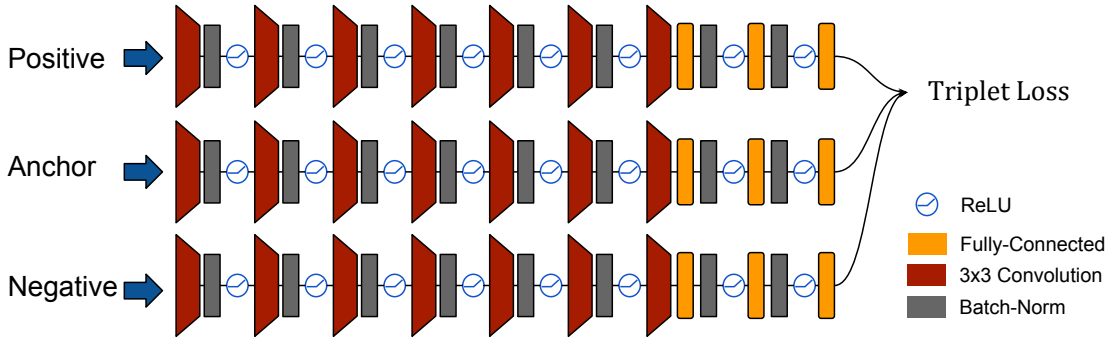


Figure 5.3: Triplet network for training.

During training, the performance of the network is assessed on various difficulty settings, and an appropriate difficulty is thus selected afterward. The difficulty settings are defined by the values we use for synthesizing patch matches, V_{pix} and V_{rot}.

Curricular training only requires us to compute the loss of the test examples without performing any back-propagation. The rationale behind this is that we do not want the harder examples exhibiting large deformations to destabilize the learning procedure and only progress when we are confident of our performance. The difficulty is automatically adjusted by either increasing or decreasing V_{pix} and V_{rot}, thus allowing the network to fall back to easier examples if it starts making mistakes on those. Although this process biases the network's performance more towards easier examples, which typically are smaller viewpoint/illumination changes, we believe this characteristic is expected of feature descriptors, as large transformations demand more explanation in a figurative sense. Algorithm 4 gives the details of the procedure.

A critical component of the training procedure is hard-negative mining. During each training iteration, we load a batch of anchors and positives, and a pool of negatives. The pool of negatives is used to choose the hardest example for each pair of anchors and positives. The hard examples are determined

through loss values exceeding a threshold. We retry mining for a fixed number of times, if no hard negatives are found, we proceed with the batch in hand.

Algorithm 4 Curricular Training. p_{step} and r_{step} determine step sizes to take in difficulty space. K specifies an upper-bound for difficulty. N defines a period of normal training. τ defines a tolerance threshold for acceptable loss difference.

function TRAIN(p_{step}, r_{step}, N, K, τ)
  $V_{pix} = 2$, $V_{rot} = 5$
  while $true$ **do**
    $PeriodLoss = 0$
    for $i = 1, N$ **do**
      $batch = LoadBatch(V_{pix}, V_{rot})$
      $loss = ForwardBackward(batch)$
      $PeriodLoss = PeriodLoss + loss$
    end for
    $AvgLoss = PeriodLoss/N$
    $V_p = 0$, $V_r = 0$
    $TestLosses = \{\}$
    for $i = 1, K$ **do**
      $V_p = V_p + p_{step}$
      $V_r = V_r + r_{step}$
      $batch = LoadBatch(V_p, V_r)$
      $loss = Forward(batch)$
      $TestLosses[i] = loss$
    end for
    $TestLosses = TestLosses - AvgLoss$
    $j = FirstIndex(TestLosses > \tau)$
    $V_{pix} = p_{step} \times j$, $V_{rot} = r_{step} \times j$
  end while
end function

5.3 Experiments

5.3.1 Training Data and Experimental Details

The training set is composed of images used to seed the hallucination procedure. We aimed to include some variance in the image contents to increase the

robustness of our descriptor. We used all images from the PASCAL VOC 2007-2012 [25] and aerial images used in Chapter 4. Our goal for mixing in aerial imagery is reflected in the affine nature of aerial images, such that they do not exhibit many 3D structure artifacts found in PASCAL VOC images, making homographies more suitable in that situation.

During training, when loading a batch, the data is generated on the fly. To reduce the effects of data-loading. We pre-processed all training images by cropping patches of size 128×128 at random and storing those for live retrieval. When a training batch is requested, we run the ORB [14] keypoint detector, discard all keypoint sizes, and randomly pick a one. The rest of the procedure is as described in Algorithm 3. All training patches were of size 64×64.

The exact network architecture is defined as follows: $C(3/16/2) - BN - ReLU - C(3/32/2) - BN - ReLU - C(3/32/1) - BN - ReLU - C(3/64/1) - BN - ReLU - C(3/64/1) - BN - ReLU - C(3/128/1) - BN - ReLU - C(3/128/1) - FC(512) - BN - ReLU - FC(256) - BN - ReLU - FC(128) - L_2Normalize$. Here $C(k/f/s)$ denotes Convolution with a kernel size of k, f filters, and a stride of s. Also, BN denotes Batch Normalization, $ReLU$ is the rectilinear activation function, and finally $FC(d)$ denotes a fully connected layer with d outputs.

We implemented our networks in Torch7 [21] and used stochastic gradient descent with an initial learning rate of 0.01, and 0.9 momentum.

5.3.2 Training with Hallucinations

In section 5.2.1, we described our method for generating synthesized matching examples. The method, by construction, allows us to change how difficult each matching pair is by respectively updating V_{pix} and V_{rot}. In Figure 5.4, we show pairs of patches with increasing difficulty. These examples show how more

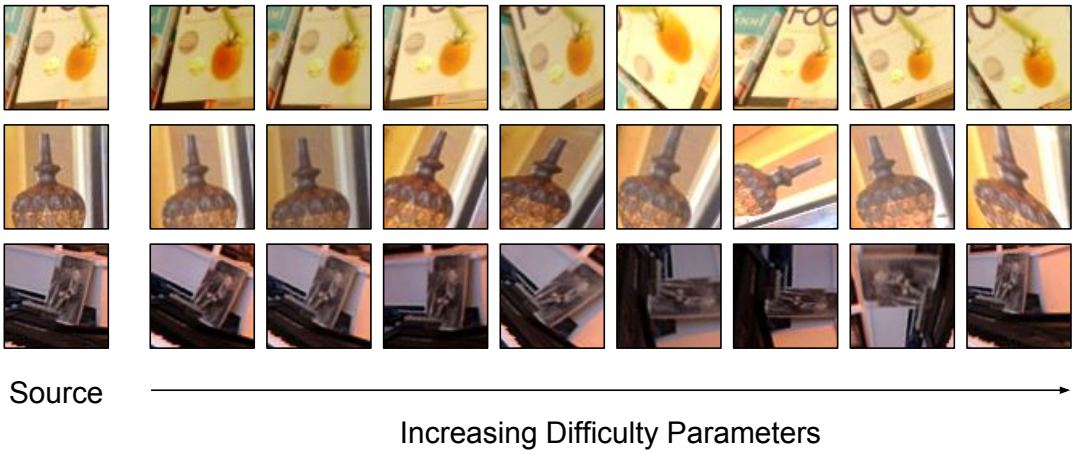


Figure 5.4: Increasing difficulty settings in data generation.

difficult examples require the network to reason more about the contents of the patch as opposed to the easy examples.

Our curricular training procedure described in section 5.2.3 should result in the live update of difficulty level depending on how the network is performing. In Figure 5.5, we see the loss function values for 8 difficulty levels, where higher numbers reflect higher difficulty. We observe an overall higher-loss for higher difficulty levels, however, at the same time we notice that easier examples do contribute to lowering the loss of more difficult examples. We attribute this to how data is generated, as under higher difficulty settings, the data generation could yield an easy example due to the random process. In general, we observe validation for our hypothesis that visually difficult examples have higher loss values.

The higher loss values of visually difficult examples go hand-in-hand with our observation of how our curricular training procedure updates difficulty. Figure 5.6 shows how difficulty levels are adjusted during training. Notice how difficulty is steadily increased and then decreased whenever the network starts performing badly on simpler examples. The figure hints at a cycle of oscillations

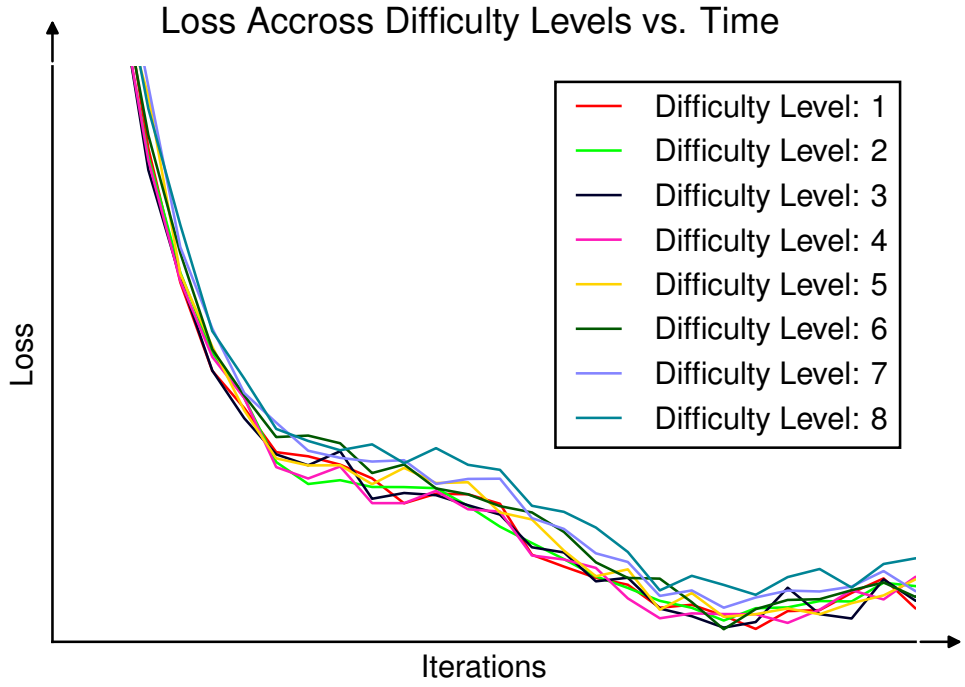


Figure 5.5: A clip of loss values across different difficulty levels during training. Higher numbers reflect higher difficulty. The plot shows more difficult examples at a higher loss level.

between progressing to hard examples and falling back to simpler ones. We believe this is due to the nature of the hardness of both being able to handle small and very large viewpoint changes. In the following subsections we will examine the network's performance across difficulty levels on synthesized examples, and then on external datasets.

5.3.3 Matching Synthesized Patches

Our first experiment is to compare our descriptor with over-the-shelf learned descriptors on synthesized patch matches. The MIT-67 indoor scenes dataset [55] was chosen to sample synthetic matches from, such that the training and testing sources are completely disjoint.

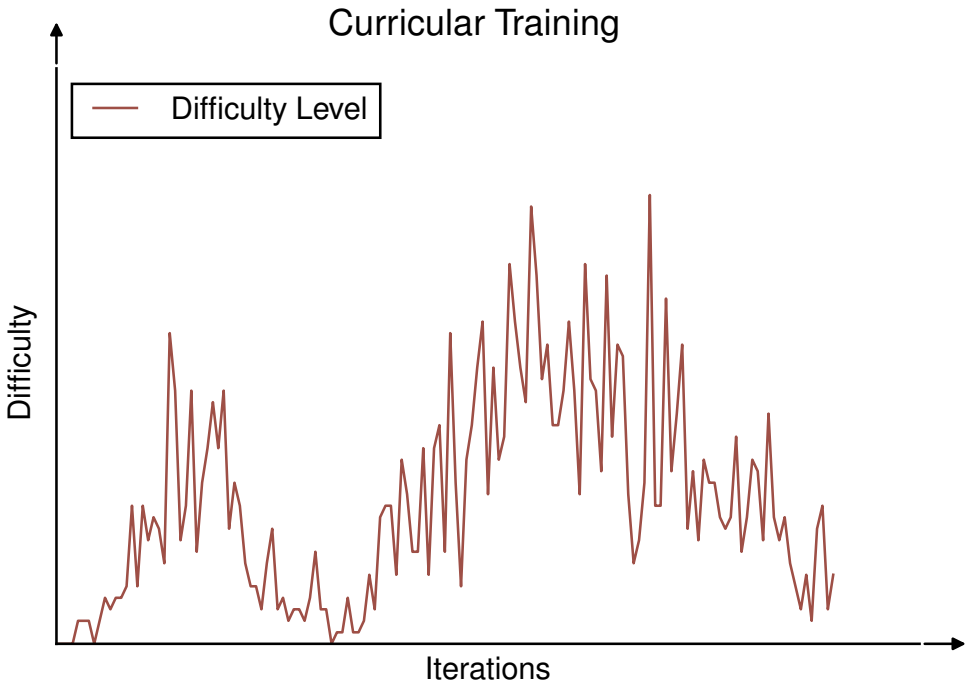


Figure 5.6: Automatic adjustment of difficulty level during training. Observe the procedure falling back to simpler examples after periods of harder examples.

The experiment is set up in a similar fashion to the sampling during training, except that we are using a different dataset to sample from. Using the hallucination method of Section 5.2.1, we generate pairs of matching patches. The goal is then to correctly identify the genuine matching pairs. In Figure 5.7, the precision/recall curves are shown for a set of patches synthesized at various difficulty settings. We interestingly observe how our method generalizes to this newly synthesized dataset without issues. Albeit this is a good indicator, the network could have learned to match hallucinated patches. This leads us to our generalizability test in the following section.

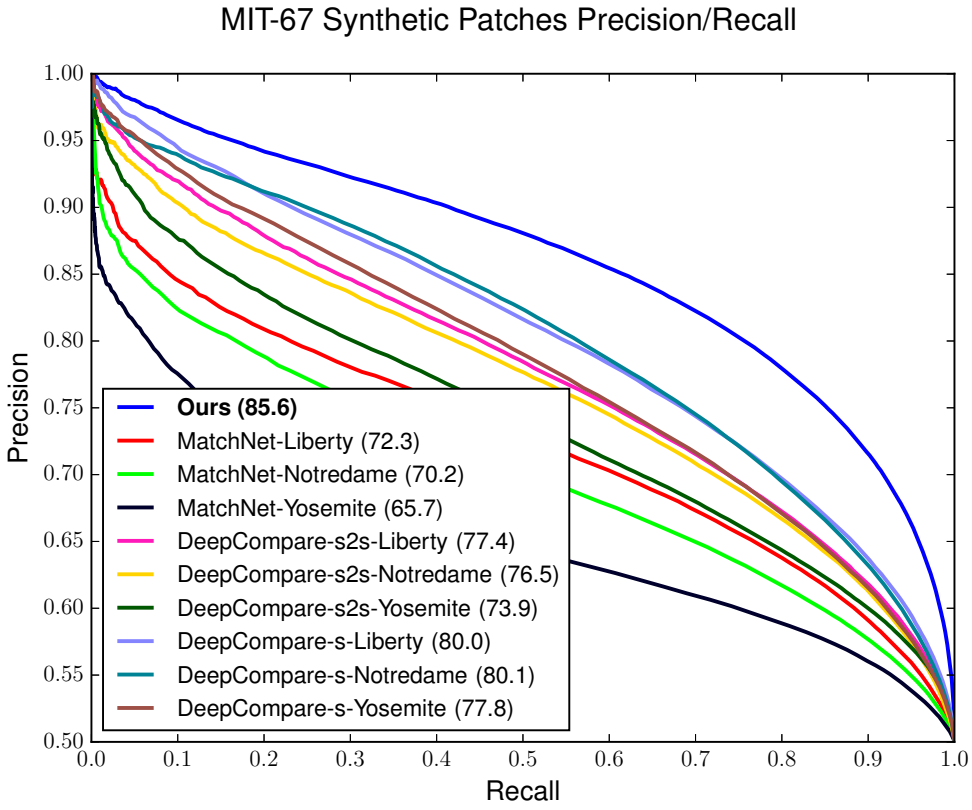


Figure 5.7: Precision/Recall Curves on MIT-67 Synthetic Patches.

5.3.4 Generalizability

HPatches

Generalizing to real world data is the genuine target task. We make use of the newly released HPatches [1] dataset to test the generalizability of our approach. We prepare two experiments for this task.

In the first experiment, we sample 256 genuine matching pairs and a pool of 255 negatives. The task is then to retrieve the genuine match out of the pool. Table 5.1 gives the results of this experiment. We would like to note that in the HPatches dataset, the keypoint detector is also used to obtain a canonical orientation for the patch, and the provided patches are cropped at the best orientation, *i.e.* without having options for de-rotated versions.

Method	Hit@1	Hit@5
MatchNet-Liberty [29]	.349	.533
MatchNet-Yosemite [29]	.191	.348
MatchNet-Notredame [29]	.354	.59
DeepCompare-s2s-Liberty [81]	.506	.661
DeepCompare-s2s-Yosemite [81]	.457	.612
DeepCompare-s2s-Notredame [81]	.475	.629
DeepCompare-s-Liberty [81]	.521	.673
DeepCompare-s-Yosemite [81]	.418	.586
DeepCompare-s-Notredame [81]	.462	.619
Ours	.520	.653

Table 5.1: Accuracy performance on HPatches.

In the second experiment, we evaluate the effectiveness of the network in a classification task and present the precision-recall results. Figure 5.8 shows how our model compares to the other baselines. We evaluated on 25k pairs. Our model's results compare favorably amongst the state-of-the-art methods. This further cements our hypothesis for the use of synthesized training data to achieve good performance.

Out of all tested methods, Deep Compare [81] is the strongest competitor, achieving 93.8% compared to 91.6% average precision. MatchNet [29] seems to exhibit slight overfitting to the training sets used. This can be seen from the retrieval experiment with top-1 retrieval scores around 35%. The encouraging aspect of evaluation is that it shows our synthesis-based descriptor performing and competing with the state-of-the-art descriptor DeepCompare [81].

In Figure 5.9, we show examples of correctly matched patches in retrieval order (from left-to-right). The transformations exhibited between the matching pairs are not very significant, but the quality of the image varies significantly, which is an artifact of the capture process. We also note how the network retrieves visually similar patches quite well.

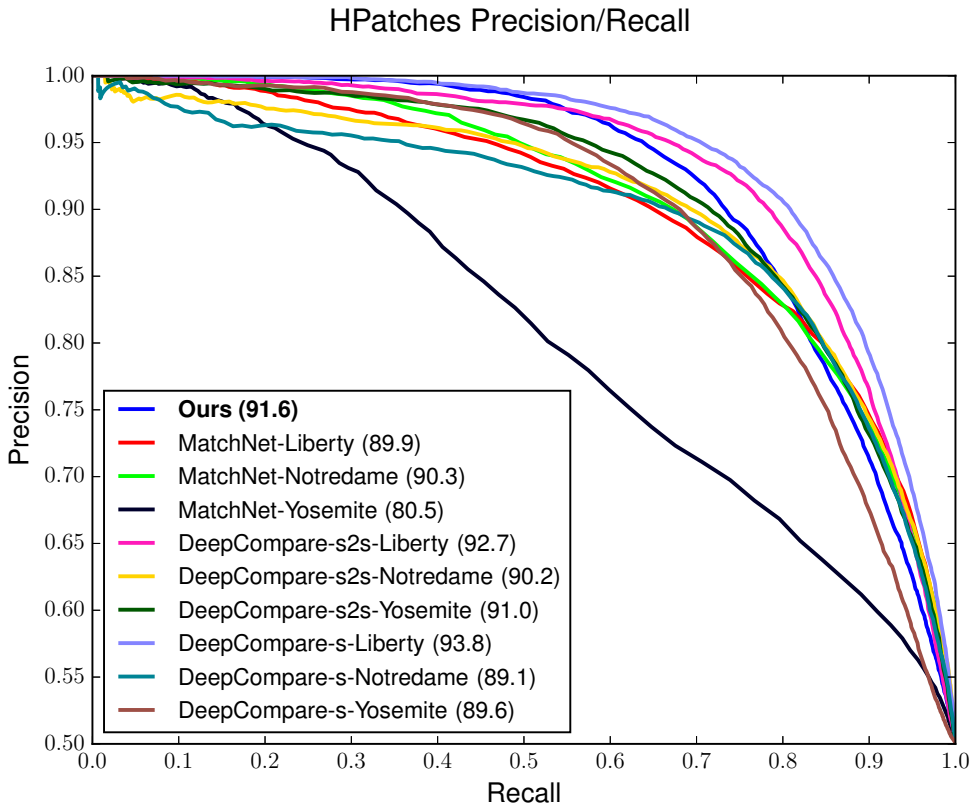


Figure 5.8: Precision Recall Curves for HPatches. The numbers between parenthesis represent the average precision.

In Figure 5.10, patches which the network was unable to retrieve correctly are presented. It is clear how visually similar the retrieved patches are, which explains the confusion of the network.

UBC Phototour Dataset

To further test our hypothesis, we performed further experiments on the UBC Phototour dataset [13]. The dataset is comprised of three separate sets, *Liberty*, *Yosemite*, and *Notredame* of matching patch pairs. We repeated the patch match classification experiment, and present here the precision-recall curve results on 25k pairs of patches obtained from each subset respectively. This is the dataset used by DeepCompare [81] and MatchNet [29] to learn patch descriptors. Fig-

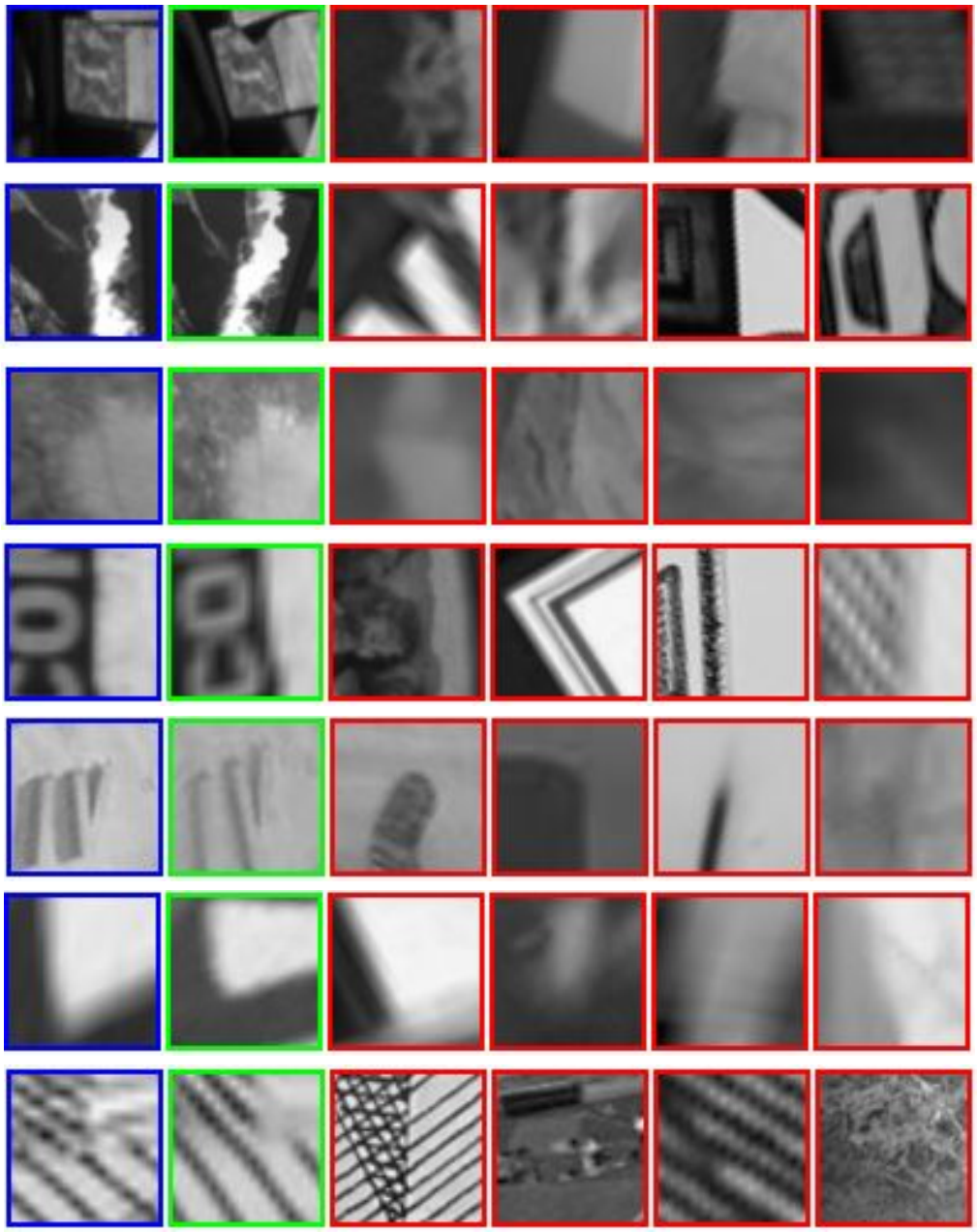


Figure 5.9: Examples of patches correctly matched from the HPatches dataset. Blue: Source patch, Green: Matched patch. Red: Incorrect patches. Order of first retrieved is left to right.

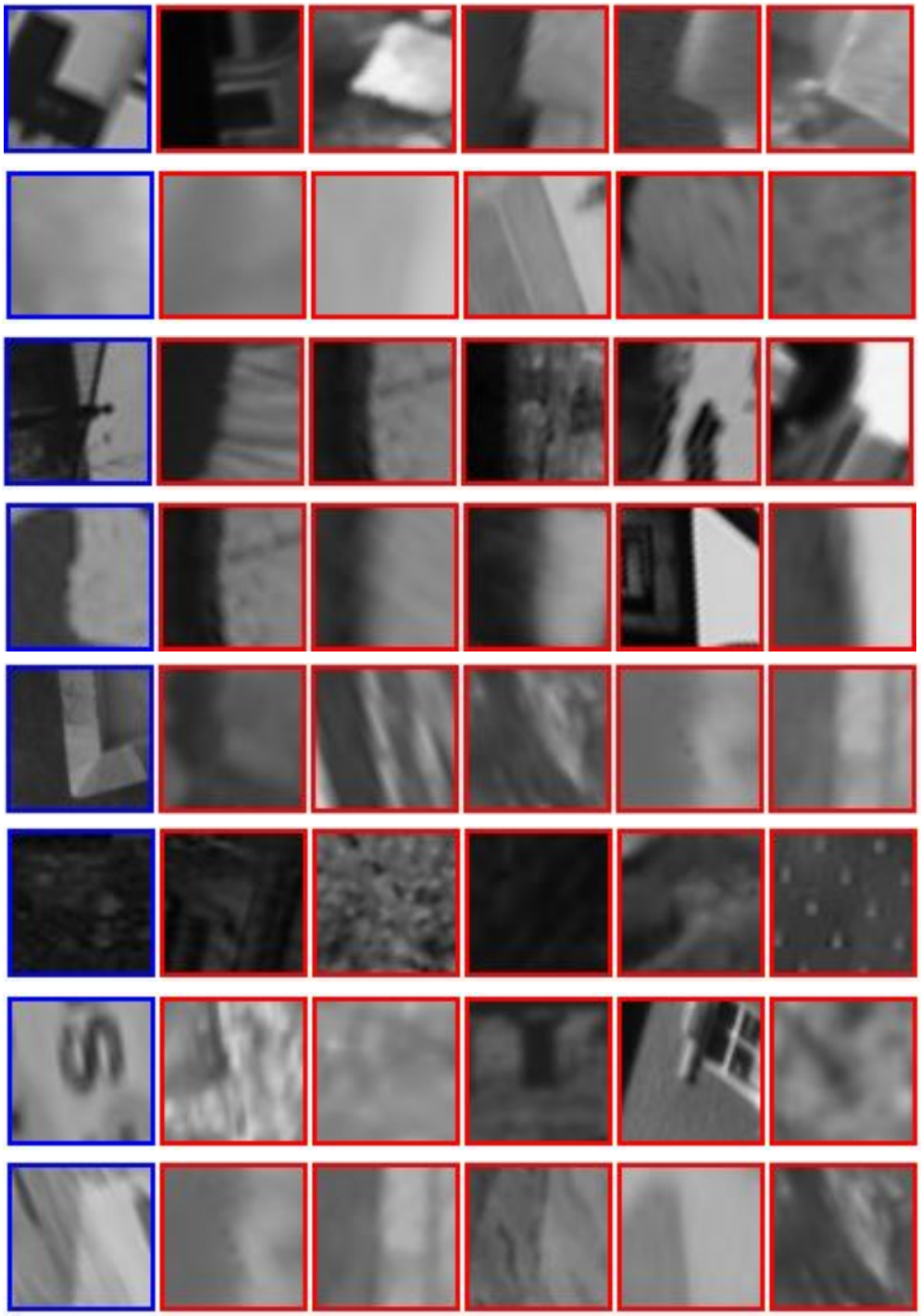


Figure 5.10: Examples of patches incorrectly retrieved from the HPatches dataset. Blue: Source patch, Green: Matched patch. Red: Incorrect patches. Order of first retrieved is left to right.

ures 5.11, 5.12, and 5.13 show the precision recall curves.

Our results are very close to the state-of-the-art methods and outperform them with the *Yosemite* subset. We believe this indicates the viability of our method, which is solely based on synthesis, and shows that improving descriptors may lie in acquiring labels for instances that cannot be synthesized easily, such as those exhibiting 3D parallax.

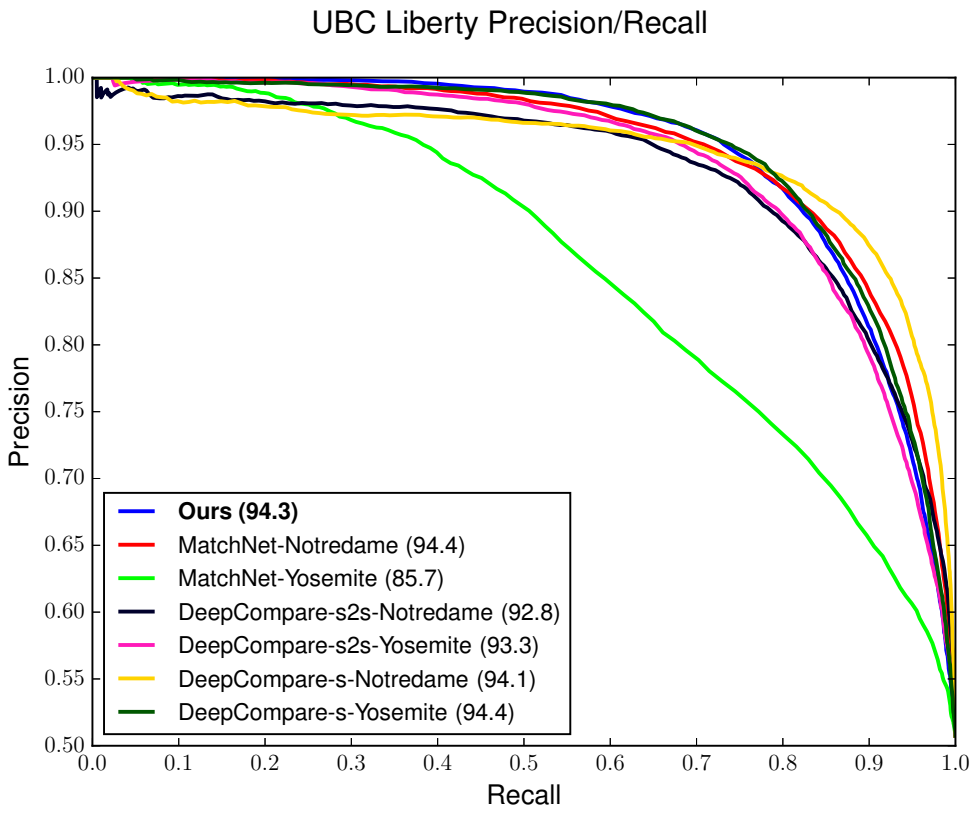


Figure 5.11: Precision Recall Curves on the Liberty subset of the UBC Phototour dataset. The numbers between parenthesis represent the average precision.

5.3.5 Discussion

An important point to note from the experimental setup and the results is that our training and test sets are from different datasets. The training set is gen-

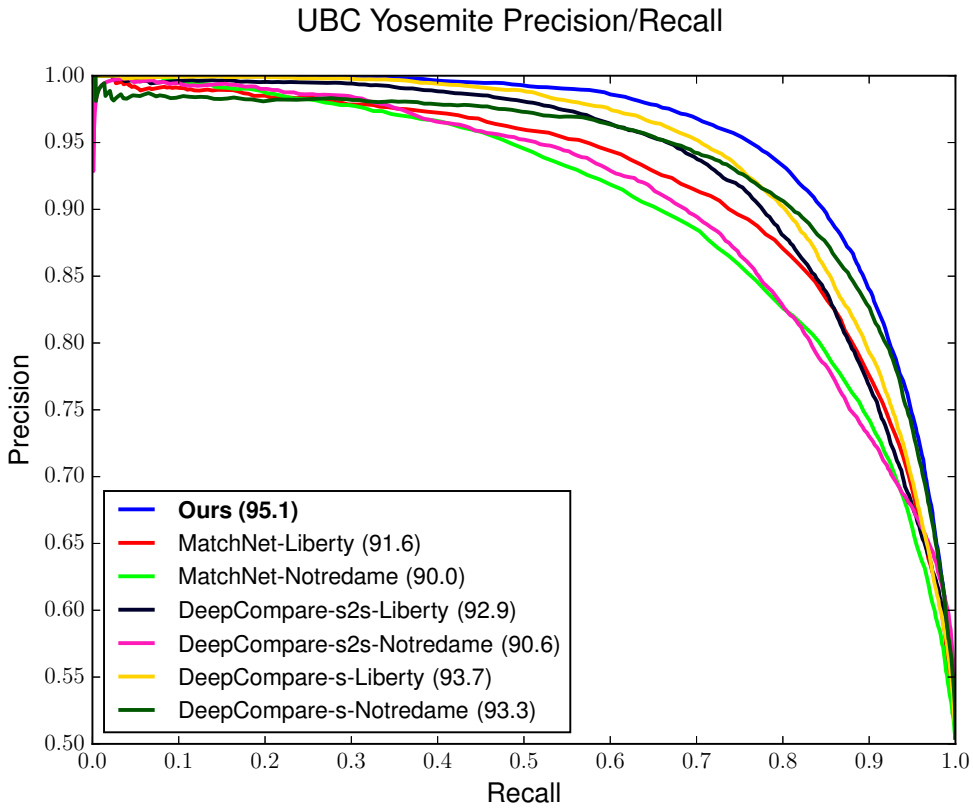


Figure 5.12: Precision Recall Curves on the Yosemite subset of the UBC Phototour dataset. The numbers between parenthesis represent the average precision.

erated from patches of the PASCAL VOC dataset (2007-2012) and aerial images used in Chapter 4 using planar homographies. On the other hand, the test datasets HPatches and UBC Phototour contain a variety of transformations. Nonetheless, the performance obtained by the presented approach is competitive with the state-of-the-art.

This demonstrates the feasibility of using *synthetic* data to obtain invariant representations that can give promising results on real data containing a larger class of transformations as shown by the results. We are being careful in not making any strong mathematical claims here. We believe that the true value of the presented approach lies in mitigating the need for high-quality real data with patch pairs exhibiting real 3D transformations between them - the con-

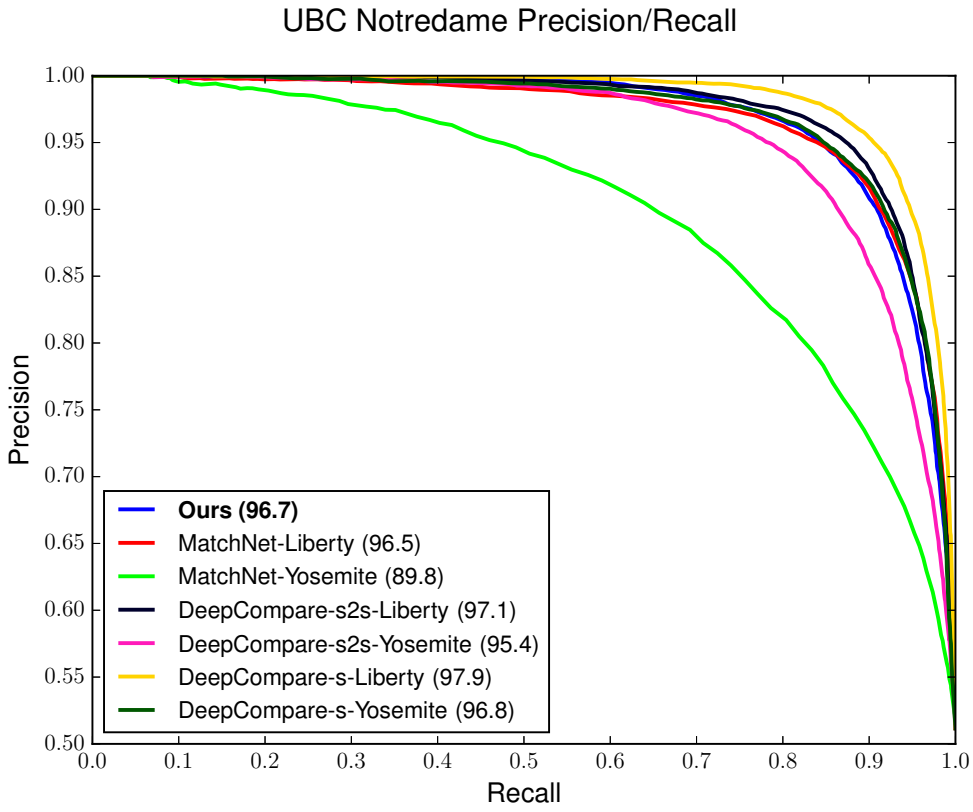


Figure 5.13: Precision Recall Curves on the Notredame subset of the UBC Phototour dataset. The numbers between parenthesis represent the average precision.

structions of the latter may impose a debilitating burden of time and resources. In particular, we think the real value of the presented approach lies in suggesting the synergistic usage of hallucinations *synthetic* data with small amounts of high-quality real data embodied with patch pairs of 3D correspondences.

5.4 Conclusion

In this chapter, we proposed a deep supervised learning based approach to compute patch descriptors. Extant approaches impose a debilitating need for the availability of large numbers of high-quality training data containing correspondences between different patches from different viewpoints. We mitigate

this need by proposing an approach whereby patches from ordinary images are used to synthesize a whole training set. The synthesis approach generates views of the same patch from different random viewpoints with the use of homographies. The synthesis approach was paired with a novel curricular learning framework that automatically guides the network through successive levels of difficulty to progressively learn *better* invariant representations. The main achievement of this chapter was to demonstrate the feasibility of this approach for learning patch representations that are shown to achieve comparable performance to state-of-the-art methods for matching patches that undergo a variety of transformations, including 3D ones.

CHAPTER 6

CONCLUSIONS

Image matching is a fundamental problem in computer vision. This thesis addressed several scenarios in wide baseline matching. The first scenario involved ultra-wide aerial pairs with the goal of finding correspondences across building facades and establishing homographies using a guided matching algorithm which relied on generating synthetic views and leveraging repeated patterns. Caveats in that approach lead to the second scenario where feature learning was introduced. In the second scenario, a large dataset of ultra-wide aerial imagery was collected with the goal of learning a model that matches images on both local and global levels using weak-supervision. The approach performed well at global matching and delivered probable local matches, however, those local matches were not verifiable due to lack of labels. The third scenario addressed that issue by converting the problem into a fully-supervised setting where convolutional neural networks are used to learn how to detect and match features. The final scenario was primarily influenced by the previous two scenarios, where the need for high-quality supervision is questioned. Using only synthetically generated training data, competitive results on several matching datasets was achieved.

This work examined several aspects and facets of the image matching problem. The use of engineered features and learned features was examined starting with correspondence matching in aerial imagery as a case study and progressed into more general correspondence matching scenarios. This line of work is still ripe for advances on many fronts. The performance of feature detectors and descriptors has room for improvement, especially in the realm of handling varying photometric and perspective deformations within the same representation.

These representations would require the development of techniques capable of capturing and encoding the various axes of variation compactly and meaningfully. Further, the development of better feature representations would go hand in hand with the development of superior matching algorithms in producing matching results that are faster to compute and higher in quality as measured by geometrical and visual correctness.

Humans, currently, vastly outperform computers in dealing with viewpoint and illumination changes. As humans appear to reason on multiple levels during a task of image matching, computer vision algorithms do not. For example, humans would leverage *objectness* and past experiences of object deformations in determining relationships between pixels. If we imagine a scene with a car and a tree photographed from two vantage points, humans do not seem to consider the relationship between pixels belonging to the car and pixels belonging to the tree in their judgment process, as they belong to separate objects and their relationship is not physically feasible. Albeit some algorithms attempt to include techniques for grouping and context analysis when dealing with pixels for matching, multi-level semantics are still missing. Researching multi-level semantics and their use in computer vision algorithms would be a promising direction to pursue.

Computer vision algorithms are slowly but surely catching up with human performance across various visual tasks, with image matching among them. Current and future research will be bridging the gap and delivering continuous improvements to the numerous applications relying on computer vision algorithms. All told, our human lives will be better at each step of the way.

BIBLIOGRAPHY

[1] Local features: State of the art, open problems and performance evaluation. *ECCV 2016 Workshop*, 2016.

[2] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. 2015.

[3] Hani Altwaijry, Andreas Veit, and Serge Belongie. Learning to Detect and Match Keypoints with Deep Architectures. In *BMVC*, 2016.

[4] Amazon.com. Amazon mechanical turk.

[5] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[6] Boris Babenko, Piotr Dollár, and Serge Belongie. Task specific local region matching. In *ICCV*, 2007.

[7] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *BMVC*, 2016.

[8] Mayank Bansal, Kostas Daniilidis, and Harpreet Sawhney. Ultra-wide baseline facade matching for geo-localization. In *ECCV 2012*.

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[10] Robert C. Bolles. Robust feature matching through maximal cliques. 1979.

[11] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994.

[13] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *PAMI*, 2011.

[14] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. BRIEF: Computing a local binary descriptor very fast. *PAMI*, 2012.

[15] M. Carcassoni and E.R. Hancock. Point pattern matching with robust spectral correspondence. In *CVPR 2000*.

[16] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[17] Ouk Choi and In So Kweon. Robust feature point matching by preserving local geometric consistency. *Computer Vision Image Understanding 2009*.

[18] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *NIPS*. 2016.

[19] Ondrej Chum and Jiri Matas. Matching with PROSAC - progressive sample consensus. In *CVPR 2005*.

[20] Yu-Chia Chung, T.X. Han, and Zhihai He. Building recognition using sketch-based representations and spectral graph matching. In *ICCV*, 2009.

[21] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[22] Microsoft Corp. Bing maps.

[23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[24] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *CoRR*, 2016.

[25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[26] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 1981.

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[28] Google Inc. Google maps.

[29] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015.

[30] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference 1988*.

[31] W. Hartmann, M. Havlena, and K. Schindler. Predicting matchability. In *CVPR*, 2014.

[32] James Hays, Marius Leordeanu, Alexei A. Efros, and Yanxi Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV 2006*.

[33] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[34] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[35] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*. 2012.

[37] David Lavine, Barbara A. Lambird, and Laveen N. Kanai. Recognition of spatial point patterns. *Pattern Recognition*, 16(3):289 – 295, 1983.

[38] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR 2005*.

[39] S. Leutenegger, M. Chli, and R.Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[40] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[41] Tsung-Yi Lin, Yin Cui, Serge Belongie, and James Hays. Learning deep representations for ground-to-aerial geolocalization. In *CVPR*, 2015.

[42] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. SIFT Flow: Dense correspondence across different scenes. 2008.

[43] Gucan Long, Laurent Kneip, Jose M. Alvarez, and Hongdong Li. Learning image matching by simply watching video. *CoRR*, 2016.

[44] Jonathan L Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *NIPS*, 2014.

[45] David G. Lowe. Object recognition from local scale-invariant features. *ICCV 1999*.

[46] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC 2002*.

[47] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 2005.

[48] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[49] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[50] Joseph L. Mundy and Andrew Zisserman, editors. *Geometric invariance in computer vision*. MIT Press, Cambridge, MA, USA, 1992.

[51] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[52] Raphael Ortiz. FREAK: Fast retina keypoint. In *CVPR*, 2012.

[53] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[54] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *ICCV 1998*.

[55] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[56] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014.

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958.

[59] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[60] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *CVPR 2007*.

[61] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015.

[62] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014.

[63] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV 2012*.

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[65] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[66] Dennis Tell and Stefan Carlsson. Combining appearance and topology for wide baseline matching. In *ECCV 2002*.

[67] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *PAMI*, 2010.

[68] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[69] Eduard Trulls, Iasonas Kokkinos, Alberto Sanfeliu, and Francesc Moreno-Noguer. Dense segmentation-aware descriptors. In *CVPR*, 2013.

[70] Tinne Tuytelaars and Luc Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC 2000*.

[71] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010.

[72] Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. TILDE: A temporally invariant learned DEtector. In *CVPR*, 2015.

[73] T. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *ISPA 2001*.

[74] Felix von Hundelshausen and Rahul Sukthankar. D-nets: Beyond patch-based image descriptors. In *CVPR 2012*.

[75] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[76] Changchang Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013.

[77] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *CVPR*, 2011.

[78] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[79] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016.

[80] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line 2011*.

[81] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *CVPR*, 2015.

[82] Amir R Zamir, Pulkit Agrawal, Tilman Wekel, Jitendra Malik, and Silvio Savarese. Generic 3d representations via pose estimation and matching. In *ECCV*, 2016.

[83] Wei Zhang and Jana Kosecka. Generalized RANSAC framework for relaxed correspondence problems. In *3DPVT 2006*.

[84] Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. Image retrieval with geometry-preserving visual phrases. In *CVPR*, 2009.

[85] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*. 2014.

[86] Marco Zuliani, Charles S. Kenney, and B. S. Manjunath. The multiRANSAC algorithm and its application to detect planar homographies. In *ICIP 2005*.